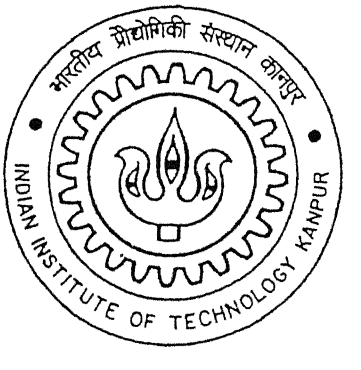
SYSTEM LEVEL SIMULATION OF ANALOG/ MIXED SIGNAL CIRCUITS USING VHDL

by

P.B.K. MURALI KRISHNA



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLGOY, KANPUR

February 2000

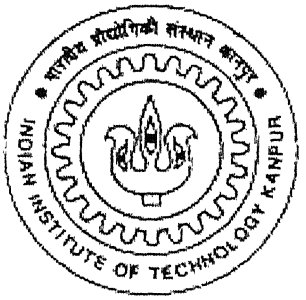
SYSTEM LEVEL SIMULATION OF ANALOG/ MIXED SIGNAL CIRCUITS USING VHDL

A thesis submitted
in partial fulfillment of the requirements
for the degree of

MASTER OF TECHNOLOGY

by

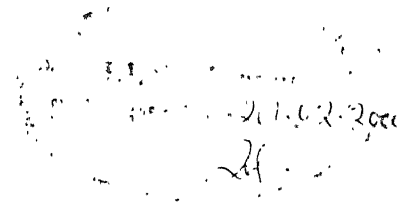
P.B.K. MURALI KRISHNA



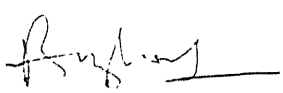
to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
FEBRUARY 2000

CERTIFICATE



This is to certify that the work contained in the thesis entitled "**SYSTEM LEVEL SIMULATION OF ANALOG/MIXED SIGNAL CIRCUITS USING VHDL**" by **P.B.K. Murali Krishna** has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.



Dr. BAQUER MAZHARI
Assistant Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur.

February 2000.

Dedicated to

My Parents

Acknowledgements

I express my deepest sense of gratitude and appreciation to my thesis supervisor Dr. B. Mazhari for his guidance, encouragement and the freedom, which he gave, is responsible for this piece of work. His ideas, and thoughts at times when this work went into troubles, have enabled me to successfully complete this work. His criticisms have improved the clarity of presentation of this thesis.

I am grateful to Dr. R. Sharan for his moral guidance and for the contacts I had with him during the course work, which developed me in the skills required for carrying out the present work. I also thank my teachers Dr. A.K.Dutta, Dr S.Kar who have taught me various courses during my academic program. I am also thankful to my friends especially Sudheer, Divyesh, Shankar, Pav, Swaroop, Murthy, Boddu, Tangal, sollu, Guduru and many other friends. Words are unable to express my feelings towards my friends who always gave me support and assisted actually throughout my stay at IIT-Kanpur.

It is my great pleasure to express my love towards my brother chakri, sister aruna and brother-in-law apparao for their constant encouragement and motivation throughout many phases of my life. Luv to sai & venu.

P.B.K. Murali Krishna

ABSTRACT

Top-Down design of complex circuits proceeds in a series of steps from the specification to the layout level. After completion of design, it is important to verify the functionality and various performance indices. The verification is most accurate if it is carried out through simulation at the transistor schematic level. However, this approach becomes untolerable as circuit complexity increases due to large simulation times. As a result, complete system simulation has to be carried out at a higher level of abstraction. This approach, for example is taken in digital systems where both functionality and timing are verified at the gate level and not at transistor level. Such an approach, although needed, is not common in analog circuits due to several reasons including absence of well-defined abstraction levels, large number of performance indices, and until recently lack of a standard language for higher level modeling of analog circuits. This thesis illustrates complete system simulation using the higher level modeling features of a hardware description language. It takes a Serial bit Switched Capacitor Successive Approximation Analog-to-Digital Converter as an example and uses an abstraction level that is one level higher than that of OP-Amps, Comparators and transistor switches, to carry out complete Analog-to-Digital converter simulation in very short times.

CONTENTS

List of Figures

CHAPTER-1

INTRODUCTION

1.1 BACKGROUND

Why Analog/Mixed Signal

Prior to 1970's, electronic circuits and systems were designed almost exclusively using analog design techniques implemented with discrete components. With the advent of integrated circuits, the digital circuits have become the basis of many systems of today, but now due to the rapid development in communications, computer and video technologies as well as other high performance applications involves with an intermingling of analog and digital behavior.

The world is now looking for mixed signal circuits where we need to integrate both analog and digital elements into a single chip for many of the robust applications. The need for analog behavior study doesn't arise just only for mixed signal design but comes into picture while interfacing digital processing applications and high performance systems.

The fabrication of complex digital systems uses sub-micron technologies in order to meet critical performance requirements such as area or speed. These systems mostly exhibit analog behavior as frequency rates are increasing and cell dimensions are decreasing. Particularly cell interconnections are behaving more and more as transmission lines with decrease in sizes and increase in frequency of operation. The integrated circuits also include A/D and D/A converters, phase locked loops, current mirrors etc. In certain areas like signal processing functions, filtering, neural applications the digital functions are replaced by an equivalent more powerful analog ones. These lead to the importance in considering the effect of analog behavior in the circuits.

***Why HDL ? It's role in Digital Design*:**

Many Hardware Description Languages (HDL) came into existence in the early 80's mainly to remove the problems faced in large design, and to cut short the design time. These HDL's are now of much more use as it does makes life easier to design complex circuits consisting of more than one million transistors. The advantages involved in using a Hardware Description Language for digital design purpose are illustrated as below:

- Hardware Description Languages like VHDL, Verilog have filled the communication gap between a vendors and customers as this is the only known best way to document the design.
- HDL's do solve problems faced in development, exchange and documentation

of digital hardware. For e.g.: a VLSI company has designed and manufactured a chip and sold to other company which is need of that chip, then that company has to understand thousands of documentation pages to know complete functionality and for the maintenance of the chip. If there is any fault then it has to diagnosis the errors and to know the behavior we need to know a language, which would do this documentation perfectly and also used for the compilation of the document.

- With the use of Hardware Description Languages we can simulate the block diagram approach using structural descriptions, as practically in the initial stages we formulate a block diagram and then approach hierarchically to final design.
- One of the greatest advantages of designing a digital circuit using a HDL is the designer can concentrate on function, i.e. implementing the required specifications and does not need to devote time and energy to technology-specific factors which do not affect the function as HDL's are technology independent.
- Complex designs could able to be handled easily by using Hardware Description Languages as these HDL's support for modeling the system hierarchically i.e. a digital system can be modeled as a set of interconnected components, each component in turn can be modeled as a set of interconnected sub-components and so on. HDL's also support top-down, bottom-up and meet-in-the-middle methodologies.
- Design times are drastically reduced using Hardware Description Languages as HDL's could handle complex problems very easily.
- Recycability is increased as the design documentation described using Hardware Description Languages were portable and hence to design a system we need not start from the scratch, instead we can make use of the earlier designs and extend the design for our needs.

***Advantages of AHDL over SPICE*:**

With the use of SPICE we get a high degree of accuracy but at a price of a relatively low component limit and long run times. The features of analog HDL, for example VHDL-AMS, were discussed briefly in Appendix-II. The advantages of using analog HDL rather than SPICE are illustrated as below:

- For a SPICE model, number of standard components are required to achieve desired function where as in a standard analog HDL implementation a simpler mathematical description is sufficient.
- Higher level descriptions using behavioral models become necessary, as they allow for the simulation of the whole system in a shorter time than an equivalent transistor level description would do.
- Using analog HDL lets you design at a higher level of abstraction. High level design makes more efficient use of your simulation resources and lets you make engineering trade-off decisions earlier in the design.
- In fault analysis of circuits some faults cause difficulties in being analyzed when using SPICE analysis as in the SPICE the calculations are all done using network of matrixes, so a fault may cause the network matrix to tend towards singularity and thus that particular fault cannot be analyzed. This will not be the case with analog HDL as here we can describe the system by behavioral equations.
- SPICE is computationally intensive and requires too much run time to reproduce results, as compared to analog HDL. Mr. Ken Kundert, cadence's design's simulation guru has quoted that " *A typical current – generation mixed signal circuit with 100,000 digital gates and 10,000 analog transistors would take 8 hours on a work station, while compared to that of EDA tools that return a response in less than 3 minutes"* [16]. The designer's idea is to try out his ideas on a "what if" basis and to see quickly the results of an incremental change. So if the designer uses SPICE it takes many days to study the performance with change in parameters, which is not the case if using an analog HDL.

- The primary difference between standard versions of SPICE and analog HDL's is that analog HDL language definitions contain many more primitives than that of SPICE and thus makes creating behavioral models easy and direct.
- Designing with a top-down design methodology is necessary for high complex mixed-signal chips, such as those in communications and multimedia systems. Analog HDL supports top-down approach of design i.e. we can start with a simple block diagram of a design and add details to it as we refine the design, progressing from concept to implementation, finally as we begin to implement the design we can work at the macro-model and eventually at transistor level, as appropriate for the design. SPICE supports bottom-up design but doesn't support top-down approach; as such complex large systems cannot be modeled or designed using SPICE.
- System Level Simulation can be done with much more ease using an analog HDL compared to that of using SPICE. In System Level simulation we need not specify all the parameters instead specify the parameters that we are interested with and describe some parts of circuit in more abstract manner and some parts detailly. For System Simulation using SPICE, all details are incorporated into it, which makes its computation times longer.
- Analog HDL and behavioral modeling encourages us to work at different levels of models, which is not possible using SPICE.
- SPICE models are accurate for some parameters but not for others, for e.g. if we are interested in temperature coefficient of an OP-Amp's offset voltage, we may find typical SPICE model poorly represents the real device. We can build an accurate behavioral model of the offset voltage .vs. temperature coefficient by using behavioral modeling if we know the behavior of OP-Amp we are simulating.
- Macro modeling can be done in SPICE but these macro models are all based around controlled sources like simple linear function of one voltage or current or polynomial functions of several voltages or currents. Such elements are therefore not sufficient to model voltage or current limiting as happens for instance when an OP-Amp's output voltage saturates. Such effects instead are modeled by inclusion of diodes. Therefore characterization of such macro models of circuits such as OP-Amp's,

Phase Locked Loop's etc. are not an easier task. This macro modeling can be done with very less effort if we use an analog HDL.

- SPICE macro models have been used successfully for some time in creating faster simulating models of OP-Amp's and other analog building blocks. The largest obstacle to improving simulation speed with SPICE macro models is probably the limited set of primitives available.

1.2 OBJECTIVE OF THESIS

The objective of the thesis is to illustrate the idea that for system level simulation of analog/mixed signal circuits we need to use some abstraction methodology similar to that been used for digital systems. Here we do also bring out the advantages in using analog hardware description languages, and how their robust features solve many problems that are faced earlier without use of analog HDL.

To illustrate the above ideas we have chosen a Switched-Capacitor Successive Approximation Analog-to-Digital Converter. As due to non-availability of AHDL we have adapted VHDL (Very high speed IC Hardware Description Language) for analog modeling. As VHDL is used for modeling only digital devices we tried to use one of the methods adapted for modeling analog devices using VHDL.

1.3 ORGANIZATION OF THE WORK

The need for an analog Hardware description Language and it's use in the analog/mixed signal domain were discussed in the present chapter. The next chapter describes the conceptual framework of the work which involves the general system level design methodology, and the problems in performing system simulation in analog and digital domain, and how they are rectified in digital domain. In the same chapter the abstraction methodology that we have proposed for analog/mixed signal circuits is illustrated briefly. In chapter-3 of implementation and results, we had discussed the full system design flow of a serial bit Successive Approximation Analog-to-Digital

Converter, and the abstraction methodology for system simulation of ADC. In the final chapter, it gives the conclusions of the work. The features of VHDL are given briefly in appendix-I, and the features of the most newly standardized analog HDL, VHDL-AMS are given in detail in appendix-II as a scope for the reader to understand the versatile modeling features that are adapted in it for analog/mixed signal domain. In the appendix-III the detailed analysis of the DAC is given for each state.

CHAPTER-2

CONCEPTUAL FRAMEWORK

2.1 BACKGROUND

The design of an Integrated circuit involves three broad steps namely specification, implementation and verification. As a result of continuing exponential in circuit complexity, the methodologies for carrying out these activities have also changed considerably. On the one hand, concepts such as hierarchy, abstraction, regularity and design reuse have been used to simplify the design process, on the other hand, increasingly sophisticated CAD tools have been developed and used for relieving the burden of synthesis and verification on the designer.

The specifications can be transformed into a transistor schematic in one step for simple circuits, while for complicated circuits consisting of thousands of transistors; this process has to be carried out in a series of steps. The design flow can be either from top to bottom or bottom to top. Although the top-down methodology is the preferred one, in practice, a mixture of both the top-down and bottom-up methodologies in the form of a meet-in-the-middle approach has to be taken.

2.2 SYSTEM DESIGN METHODOLOGY

Fig:1 describes System design methodology. The first step involves selection of an algorithm from a variety of available algorithms to meet the specifications or if an existing algorithm fails to meet the requirements, develop a new one. The algorithm, because of its abstract nature describes only in broad details how the functionality in Analog-to-Digital Converter (analog to digital conversion) may be achieved. It is very difficult to estimate the different performance characteristics of the circuit implementation of the algorithms, but a choice nevertheless has to be made between the different algorithms available so as to proceed to the next lower step in the design. This illustrates the fundamental difficulty in the top-down design process. To complete one step of a design, decisions have to be taken based on a prediction of design characteristics at lower subsequent steps. This need to predict introduces uncertainties in the design process especially when the design being undertaken is a novel one so that the past experience is not a reliable guide. This highlights the need for information flow from the bottom to the top and the often, iterative nature of the design process.

As the design flows from one step to the next, it is important to verify that each implementation is correct in terms of both the functionality and performance. The most common verification technique is through simulation of the design. For example, the correctness of the choice of the algorithm is verified by coding it in a high level language such as C, but increasingly in languages designed for hardware modeling such as AHDL for mixed analog digital circuits.

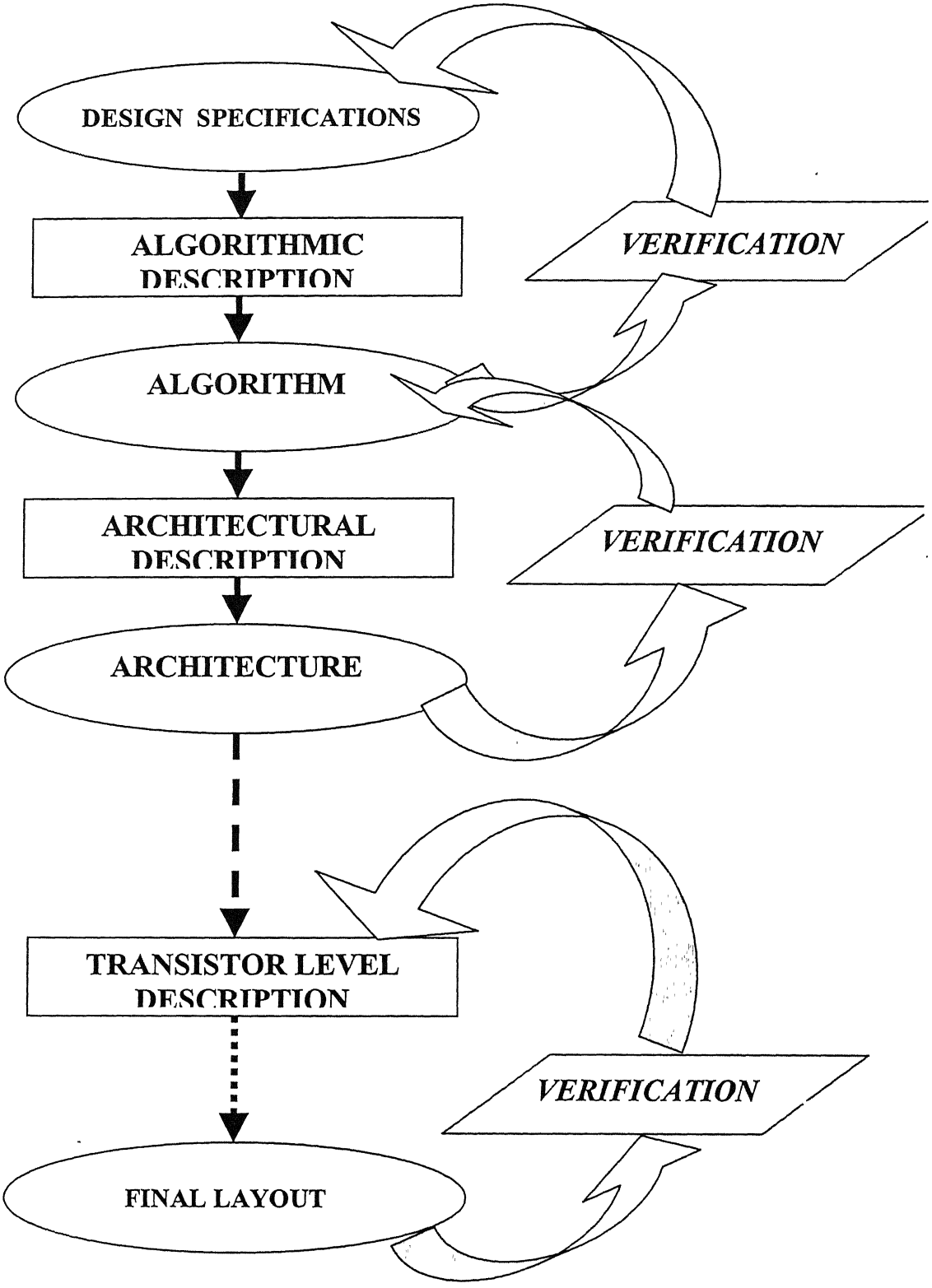


Fig: 2.1 System design methodology

Simulation vectors generated before hand are applied to the executable algorithm and the output response is checked for correctness with respect to the specifications. Once the decision regarding the algorithm is fixed, the next design step involves design of architecture for executing the algorithm. Defining architecture involves finding a suitable interconnection of appropriate blocks and fixing the specifications of the blocks. As before, different architectures are explored before settling down to the final choice. The ease with which different alternatives can be generated and examined is crucial for obtaining an optimum architecture. A hardware description language capable of versatile high level modeling can be very useful here to examine the properties of a tentative architecture and to answer "What if" kind of questions. It can also help considerably in fixing the specifications of the blocks in the architecture.

After the architecture design is finalized, its correctness is verified with respect to the algorithm again using simulation. The architecture is modeled in the HDL, simulated and its response compared against that obtained from the executable algorithm. This process of algorithm and architecture design and verification described above is repeated for different blocks of the architecture and the design is refined till a transistor level description of the whole design is obtained. As the design progresses to lower levels, its final characteristics become more and more apparent, but complete system characteristics become accessible only when transistor level schematic gets defined.

Circuit simulation of transistor level design can provide accurate information about the implementation but this has become increasingly difficult in analog circuits as complexity of circuits has increased. In digital domain, due to much larger design size, this goal of estimating system characteristics from simulation of its transistor level description has long been abandoned due to unpractically long simulation times. In the digital domain, the complexity of verification through circuit level simulation has been alleviated to some extent by carrying out verification at the gate level. By raising the level of abstraction to the logic level both the components count and behavioral complexity is sharply reduced. This is because now groups of five, ten or more transistors

are clumped together and described by a single symbol whose behavior is described not in terms of detailed voltage/current vs. time but in terms of much simpler boolean expressions and a few delay parameters.

Analogous to this approach it might be expected that raising the abstraction level of simulation could be used to reduce complexity of full system simulation in analog circuit domain also. However there are several problems. The first of them is that unlike the digital circuits where several abstraction levels such as circuit, logic, register transfer levels exists, in analog domain no such abstraction levels exist, that can be universally applied to a large no of circuit types. For example, transfer function can be used to describe a simple RC filter but it is unsuitable to describe a simple diode rectifier. Therefore, separate techniques have to be developed for different classes of circuits. It is important to point out here that by migration to gate level, complexity is reduced through use of two techniques namely hierarchy and abstraction. Hierarchy, by itself reduces component count by replacing several symbols by one symbol but does not by itself reduce the behavioral complexity of the new symbol with respect to behavioral complexity of collection of symbols it replaced. It is the process of abstraction which reduces this complexity by reducing details such as replacement of a complicated voltage vs. time characteristics by a single 0 -> 1 step. An example of use of hierarchy in analog circuits is use of a macro model for an opamp instead of its detailed structured containing 10-20 transistors. This would be an example of pure hierarchy if the macromodel mimics completed the behavior of Transistor level OP-Amp schematic.

Often, the macromodel includes only a few features of interest and is therefore an abstraction of the transistor schematic. An extreme example of abstraction in opamp is the ideal opamp model where only it's most important characteristic namely 'virtual ground' is retained. This shows that there is no single unique abstraction of an opamp but many different ones depending on the application. Even for the same application, different abstractions are necessary for different purposes. This brings out the major difference between analog and digital circuits, which prevents definition of universal abstraction levels. Analog circuits have many more performance characteristics unlike

digital circuits. An opamp for example can have as many as 20 performance parameters while an 8-bit adder has just one (or two) namely timing(and power dissipation).

2.3 OUR APPROACH

In the present work use of hierarchy and abstraction is made to obtain a simple model of an ADC so as to carry out its complete simulation. The A-D converter used is a switched capacitor circuit and the modeling strategy presented in this work could be used for many other circuits belonging to this class. Fig:2.2 below shows the A/D-circuit.

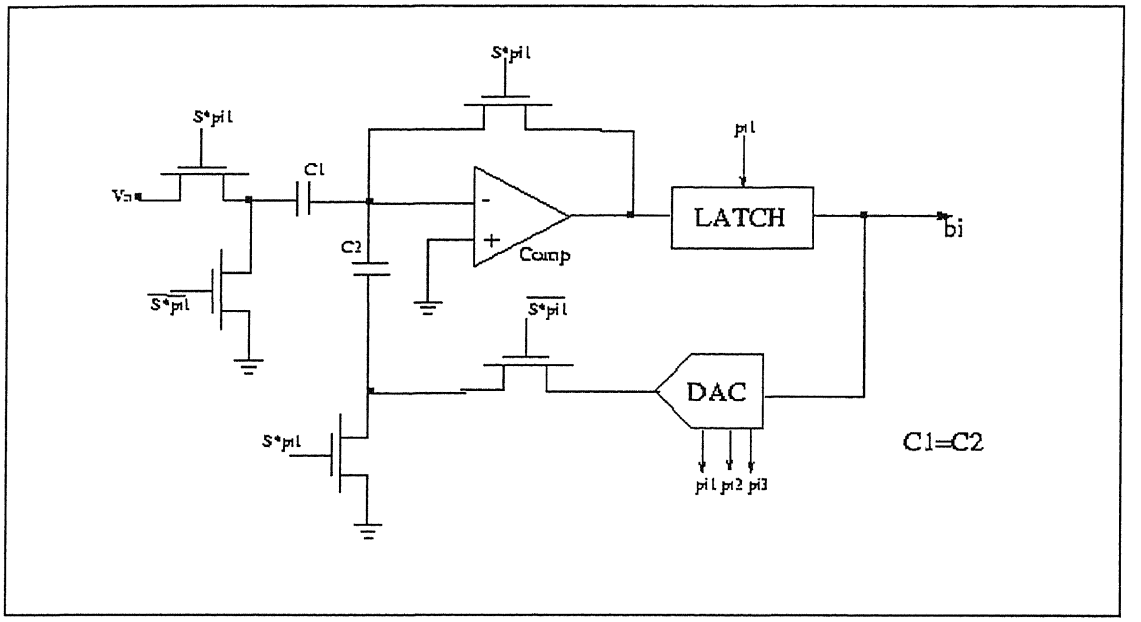


Fig:2.2. A complete architecture of ADC

The integral and differential non-linearity characteristics of the analog to digital converter were chosen for study through complete ADC simulation. For these performance characteristics, an abstraction level, which involved replacement of all transistor switches, capacitors and opamps by a single behavioral model that included effects of gain and offset voltage was used. The details of the abstraction process and results obtained are described in detailed in the next chapter. The methodology of abstraction that we have attempted for use of analog/mixed signal is shown in Fig-2.3.

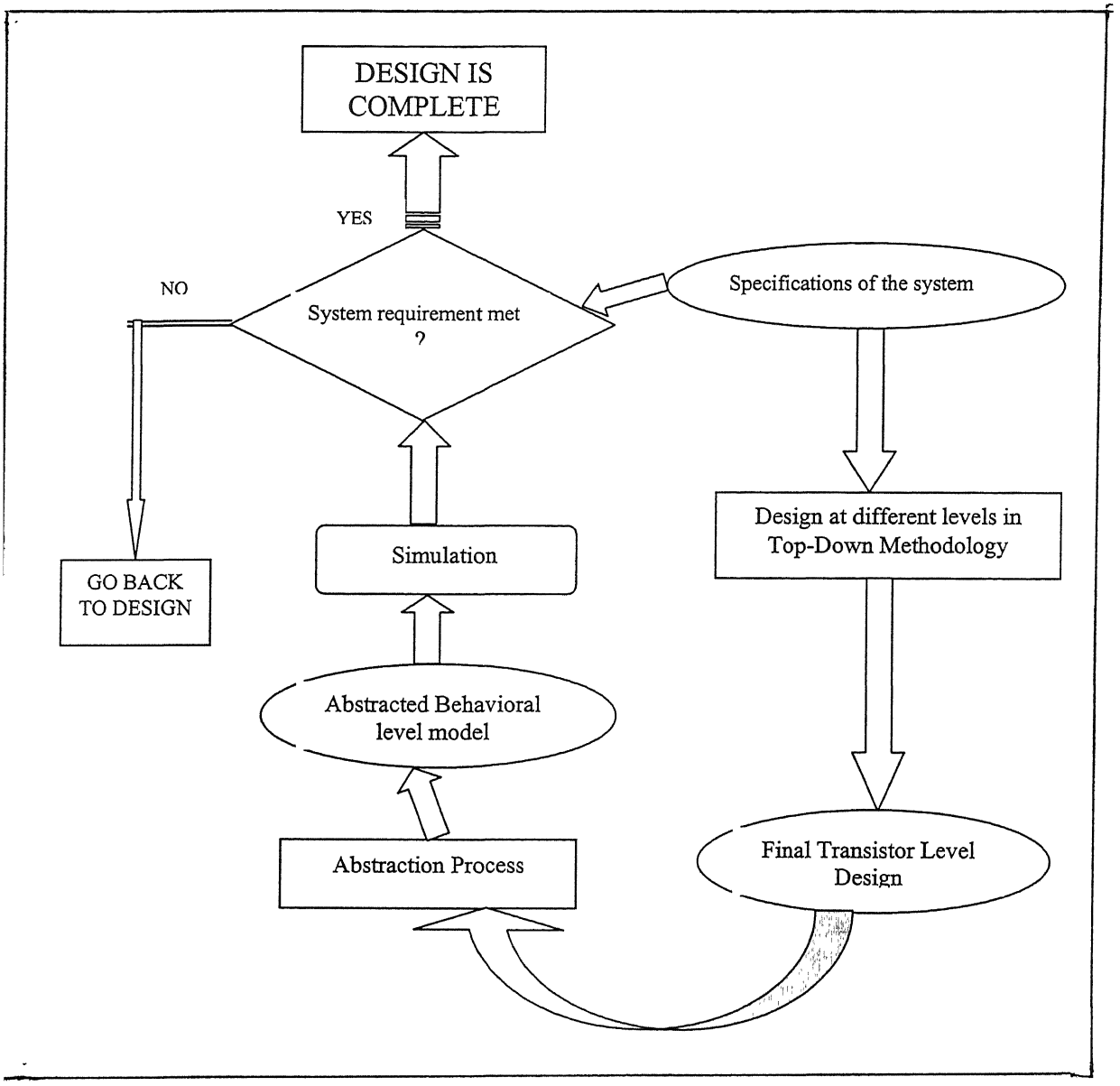


Fig: 2.3 Abstraction methodology for analog/mixed signal systems

It is interesting to point out that in the normal design flow from top to bottom, the design does pass from higher abstraction levels to lower abstraction level so it may be at first thought that an existing design abstraction level be used for full system verification also. However, the abstraction levels developed in the design process were not found adequate for this task so a new abstraction level had to be defined and used for the

purpose of verification as depicted in Fig: 2.3. This is not very surprising, keeping in mind that the requirements of design and verification process are different and may not match.

CHAPTER 3

IMPLEMENTATION & RESULTS

The Successive Approximation Analog-to-Digital Converter that we have chosen is a switched capacitor circuit. Because the behavior of switched capacitor circuit is closer to digital characteristics and because of non-availability of Analog Hardware Description Language (Analog HDL), the VHDL language was adapted to model the Analog-to-Digital converter.

SYSTEM SIMULATION OF SUCESSIVE APPROXIMATION A/D CONVERTER:

3.1 Background:

The switched capacitor successive approximation analog-to-digital converter studied in this work incorporates a serial digital-to-analog sub-converter for generating the threshold voltage sequence. In this type of converter, an input analog voltage is converted to the binary number by being compared successively with the quantatized threshold voltage sequence which, starting with a half reference voltage, increases or decreases depending on the bit value determined in the previous cycle to approach the input analog voltage.

3.2 System Level Design of ADC

3.2.1 *Algorithm for ADC*

The complete Successive approximation Analog-to-Digital Converter design is done in a series of steps. Initially an algorithm is chosen [6] to design an ADC based on various factors. The algorithm flow is shown in Fig: 3.1.

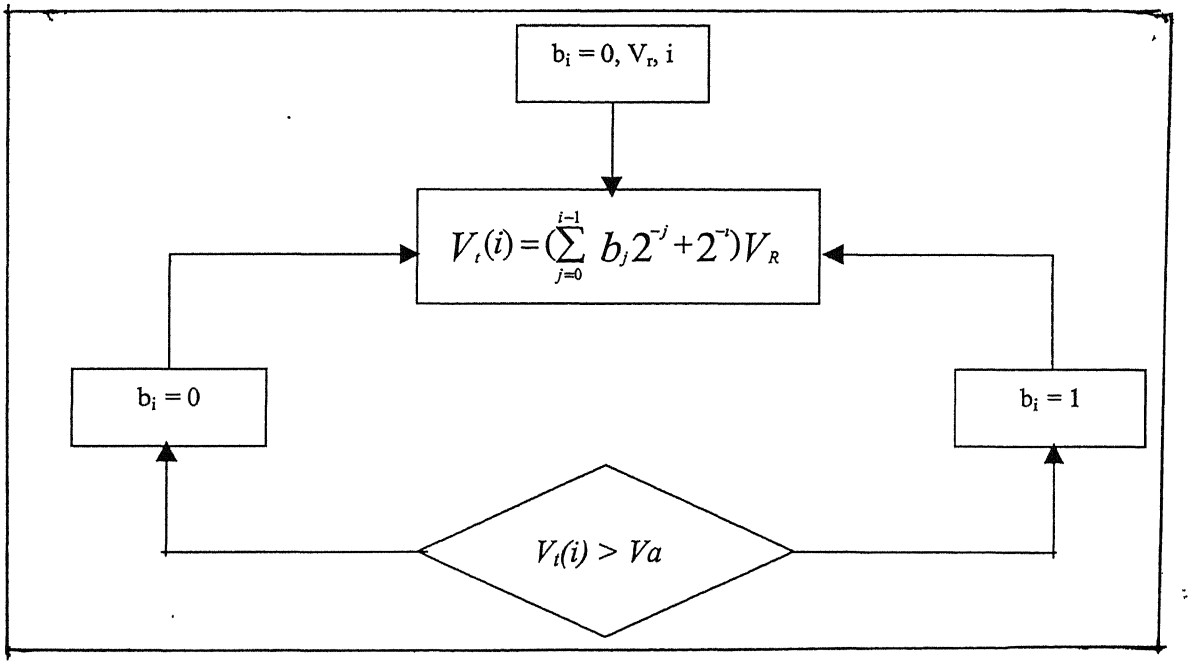


Fig: 3.1 Flow chart diagram of serial bit ADC

The value of $V_t(i)$ is obtained initially first from eq-(3.1) The value of $V_t(I)$ is obtained from eq-3.1 by initially taking the value of b_I to be zero (i.e. b_o=0) and then this $V_t(i)$ is

compared with the input analog voltage. The o/p value of the bit b_I is zero or one depending on the sign of the difference between input analog voltage and $V_t(i)$.

Algorithmic Steps:

1. The value $V_t(i)$ is calculated as given by eq-(3.1) taking the initial value of b_i to be zero (i.e. $b_0 = 0$).

$$V_t(i) = \left(\sum_{j=0}^{i-1} b_j 2^{-j} + 2^{-i}\right) V_R \qquad 3.1$$

   Where V_R = reference voltage; $b_0 = 0$;

2. If the value of $V_t(I)$ is greater than the input analog voltage i.e. $V_t(i) > V_a$ then the bit $b_I = 0$.
3. If the value of $V_t(I)$ is less than the input analog voltage i.e. $V_t(i) < V_a$ then the output bit $b_I = 1$.
4. The value of $V_t(I)$ updates to its new value according to the value of b_I, from eq-(3.1). and the process is repeated.

This algorithm can be simulated in any HDL like VHDL or a programming language like C. Although at this stage it can be done in 'C' but it is better to do simulation in HDL like VHDL as its language constructs are defined and supportive as we go to further stages in the design, This algorithmic level description was coded in VHDL and the desired response obtained is shown in Fig:3.2 .

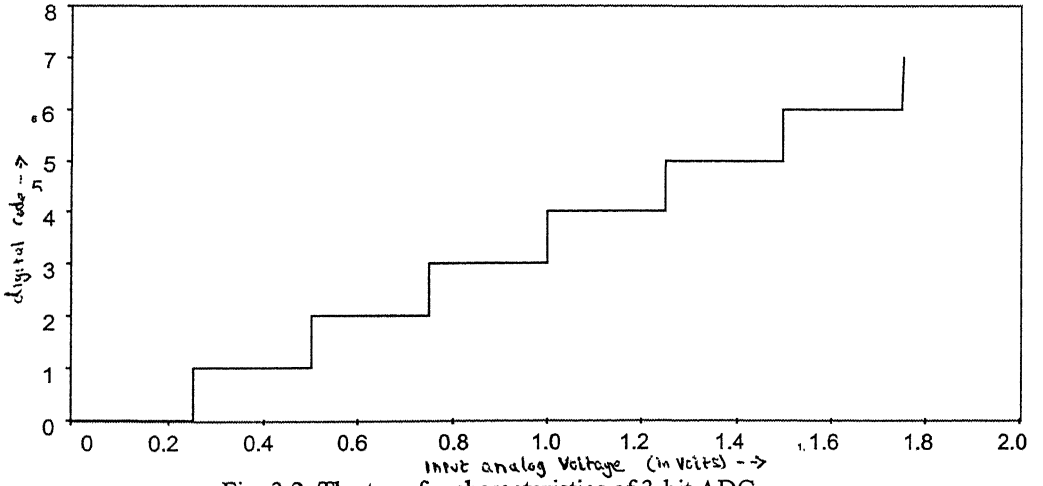


Fig: 3.2. The transfer characteristics of 3-bit ADC

3.2.2 *Architecture of ADC*

Once the decision regarding algorithm is fixed then the next step is to choose or develop an appropriate architecture that would execute the algorithm. One possible architecture[6] for the algorithm described earlier is shown in Fig:3.3.

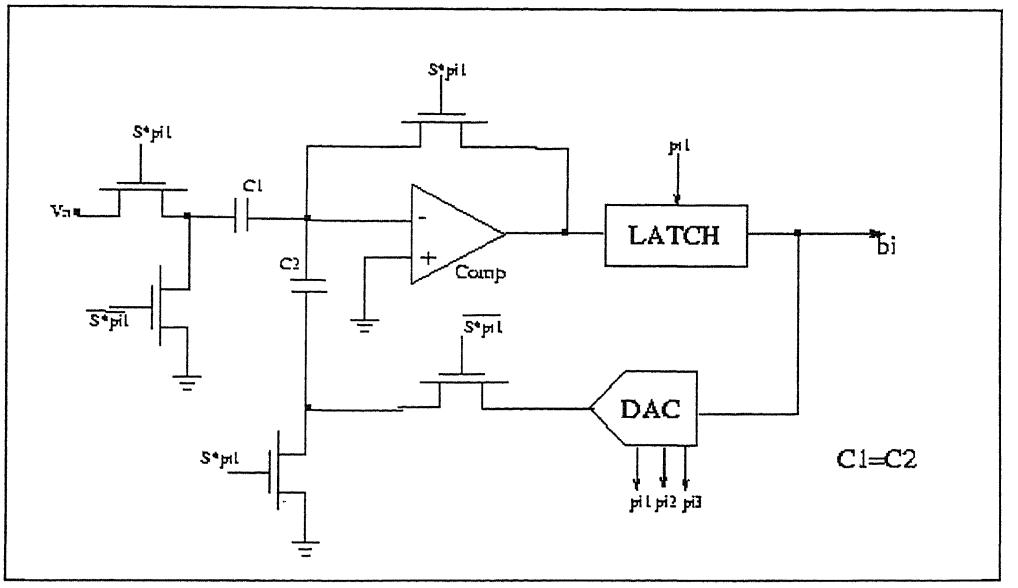


Fig:3.3 A complete architecture of ADC

The required timing diagram i.e. the clock signals to be applied for the architecture of ADC shown in Fig:3.3. is given in Fig:3.4.

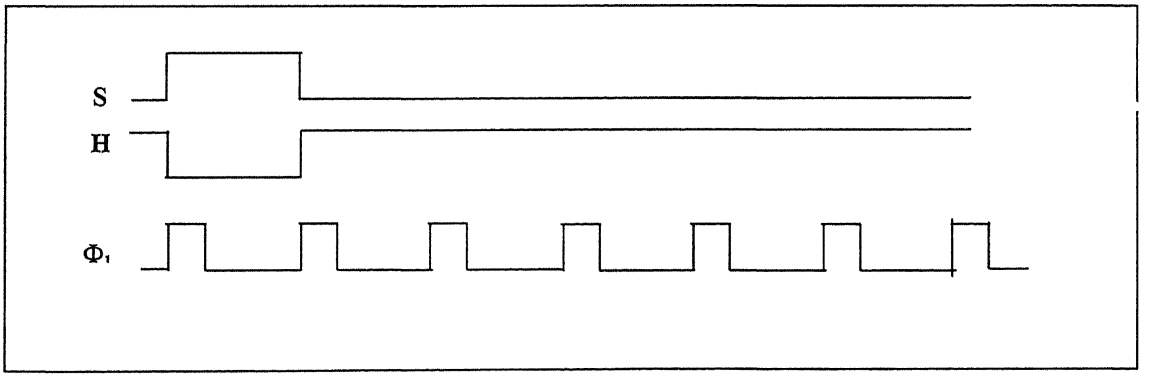


Fig:3.4 A timing diagram of digital clock signals used in architecture of ADC

The architecture of ADC contains a Digital-to-Analog Converter, a latch, a comparator, switches and capacitors as shown in Fig:3.3. Once this ADC architecture is chosen, we need to do mixed-behavioral/structural description of the whole circuit and verify the results with those obtained with algorithmic level simulation of ADC. If the results do not match with the algorithmic level simulations, then appropriate modifications to the architecture are done and again verification through simulation is repeated. This process is continued till the results match with that of algorithmic level simulations.

3.2.2.1 Basic operation of the ADC architecture

The complete architecture of ADC is shown in Fig:3.3. In ϕ_1 = '1' phase of the sample state (i.e when S=1 & ϕ_1 = 1), an input analog voltage is sampled into C1, and in the subsequent conversion state, the analog voltage thus stored in C1 is compared with the threshold voltage sequence generated by DAC, to be converted into its digital equivalent. The output of the comparator is $\pm V_{sat}$ depending upon the difference in input analog voltage and threshold voltage ($V_t(I)$). The output of latch changes to zero or one depending on the output of the comparator $+V_{sat}$ or $-V_{sat}$ respectively.

This mixed-behavioral/structural description of ADC is done by structural representation of the components like DAC, Latch, comparator, switches and capacitors. For developing the structural description of DAC we need an analog HDL, as there are capacitors, switches in the circuit, which can be modeled using VHDL-AMS (an analog HDL). Because of unavailability of analog-HDL, we have modeled using VHDL, but with slight change in the interpretation of the components like capacitors and MOS switches. For simulation of this Mixed-behavioral/structural description we need to develop before hand the behavioral description of the sub-components like DAC, Latch and comparator. The behavioral description of a Latch and comparator at abstract level is simple and straightforward, and the behavioral description of DAC is given by eq-(3.1). Once the architecture definition is verified with that obtained from algorithmic level simulation the design moves to the synthesis of sub-components like DAC, Latch and Comparator.

3.2.3 Digital-to-Analog Converter:

The next step in the design of sub-component DAC used in the mixed structural/behavioral description of ADC is again selection of a suitable algorithm that would meet the system requirements.

The output b_I of the comparator (in Fig:3.3) in each cycle indicates the voltage range to which the analog input voltage V_a belongs. For instance, assume the values of the first three bits to be $b_1b_2b_3 = (101)_B$ The most significant bit (MSB) b_1 = '1' implies that input voltage V_a is in the range from $V_R/2$ to V_R. The next MSB b_2 = '0' means that input voltage V_a is in the range from $V_R/2$ to $(3/4)V_R$ and the next MSB b_3 = '1' implies that input voltage V_a is in the range from $(5/8)V_R$ to $(3/4)V_R$. It is obvious from this example that the voltage range is successively halved, and thus algorithm given below[6] is formulated.

3.2.3.1 Algorithm for DAC:

Let the lower and upper bounds of the voltage range determined in the i-th cycle be $V_L(i)$ and $V_U(i)$ respectively. Then, $V_L(i + 1)$ and $V_U(i + 1)$ in the next (i+1)th cycle are given by eq-(3.2) and eq-(3.3) respectively.

$$V_L(i + 1) = V_L(i) + b_I\, 2^{-i}\, V_R \tag{3.2}$$

$$V_U(i + 1) = V_L(i + 1) + 2^{-i}\, V_R \tag{3.3}$$

Where $V_L(1)$ = '0' and $V_U(1) = V_R$.

The threshold voltage i.e output of digital-to-analog converter $V_T(i + 1)$ required for the next comparison is given by the following equation:

$$V_T(i + 1) = [\, V_L(i + 1) + V_U(i + 1) \,] / 2 \tag{3.4}$$

Expressing equations (3.2) to (3.4) in the recursive forms, we can obtain the algorithm for generating the threshold voltage sequence and thus the following equations (3.5) to (3.7) are obtained.

$$V_T(i+1) = [\, V_T(i) + \overline{b}_i V_L(i+1) + b_i V_U(i+1) \,] / 2 \qquad (3.5)$$

$$V_L(i+1) = b_i V_T(i) + \overline{b}_i V_L(i) \qquad (3.6)$$

$$V_U(i+1) = \overline{b}_i V_T(i) + b_i V_U(i) \qquad (3.7)$$

Where $V_T(1) = V_R/2$.

This algorithm chosen can be simulated either in HDL or a programming language like 'C' to verify that the functionality of the digital-to-analog converter by comparing with the simulation results obtained from algorithm description of DAC by eq-(3.1).

3.2.3.2 Architecture of DAC:

Once the decision regarding the algorithm is fixed the next step is to choose or develop an appropriate architecture that would execute the algorithm. From the above recursive algorithm chosen we can conclude that the digital-to-analog converter can be realized by the Sample/hold circuit and divide-by-two stage circuits. One possible architecture [6] for the algorithm described earlier is shown in Fig:3.5.

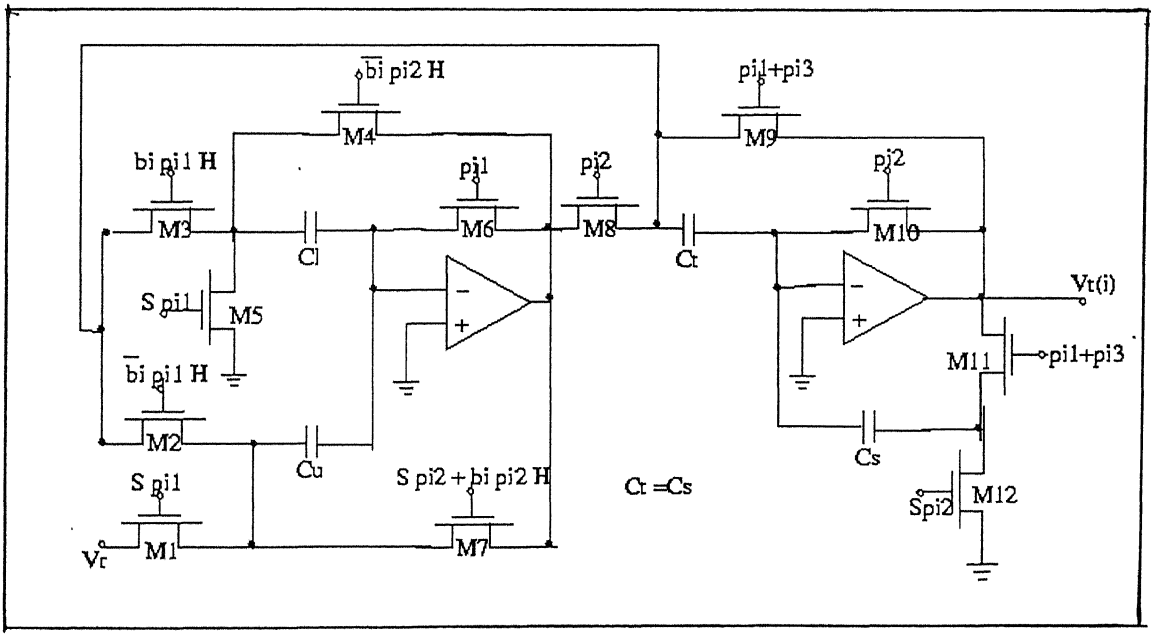


Fig:3.5. A complete architecture of Digital-to-Analog Converter (DAC)

The DAC operation is controlled by non-overlapping three-phase clocks ϕ_1, ϕ_2, ϕ_3 and the state signal 'S' which distinguishes between the sample and conversion states. The output bit b_I is invalid bit during the time period for, which the sample signal is high, or hold signal low. The timings of ϕ_1, ϕ_2, ϕ_3, S and H signals are shown in Fig:3.6.

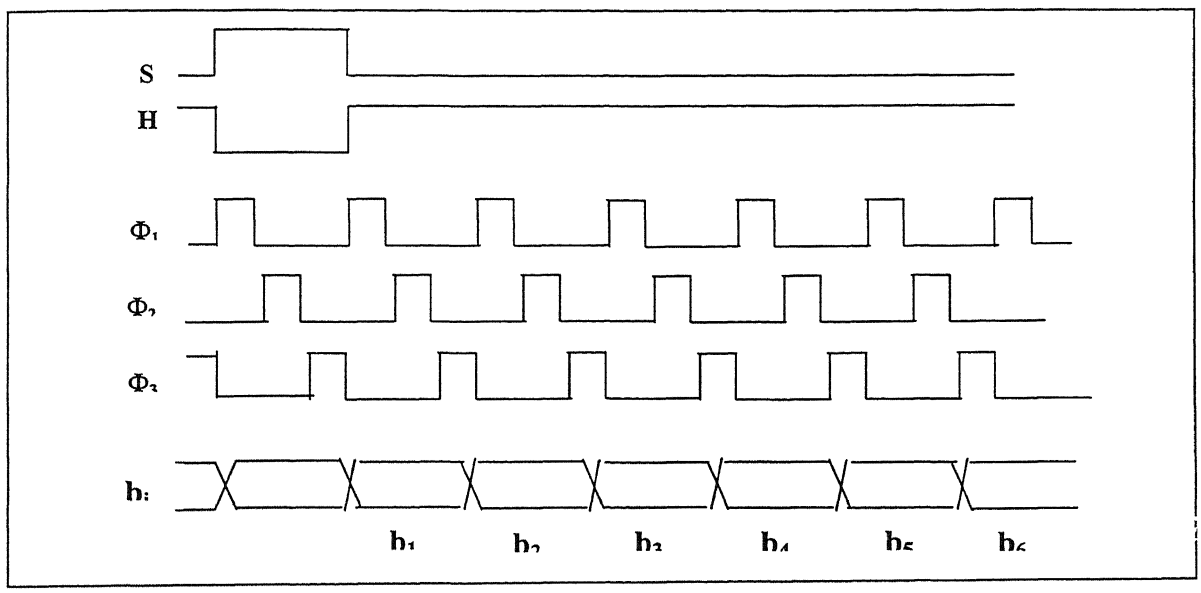


Fig:3.6 A timing diagram of digital clock signals used in the architecture of DAC

As shown in Fig:3.5, after obtaining complete architecture of DAC in terms of sub-components like OP-Amp, switches and comparators, we need to verify the functionality of the architecture obtained. Once this architecture is chosen the mixed-behavioral/structural description of the whole circuit is done and then the results are verified with those obtained at detailed algorithmic level simulation of DAC. If the results do not match with the detailed algorithmic level simulations, then appropriate modifications, in the chosen architecture is done and simulations are repeated.

3.2.3.3 Basic operation of the DAC architecture:

The complete architecture of DAC is shown in Fig:3.5. In this DAC architecture, at the i-th cycle of conversion state, three capacitors C_L, C_U and C_T store the lower bound $V_L(i)$, upper bound $V_U(i)$ and threshold voltage $V_T(i)$ sequences respectively. C_T is a unit capacitor matched to C_S. In the ϕ_1 = '1' phase of the next cycle, either $V_L(i)$ or $V_U(i)$ is

updated to $V_T(i)$ depending on the bit value b_i determined by the previous conversion. In the next ϕ_2 = '1' phase, C_T is charged to the new $V_L(i + 1)$ or $V_U(i + 1)$. In the subsequent ϕ_3 = '1' phase, C_T and C_S are connected in parallel to perform the divide-by-2 operation

This mixed-behavioral/structural description of DAC is done by structural representation of the components like OP-Amps, switches and capacitors. For simulation of this Mixed-behavioral/structural description we need to develop before hand the behavioral description of sub-components like OP-Amp. This Mixed-behavioral/structural description is best done using an analog Hardware Description Language. After the functionality is verified with that obtained from detailed algorithmic level simulation, we need to design sub-components like OP-Amp seperately in detail.

3.2.4 **OP-Amp:**

The behavioral description of sub-component OP-Amp used in the mixed structural/behavioral description of DAC serves as the specification when considering the detailed design of the sub component OP-Amp. A suitable architecture for OP-Amp [6] is selected or developed as shown in Fig:3.7.

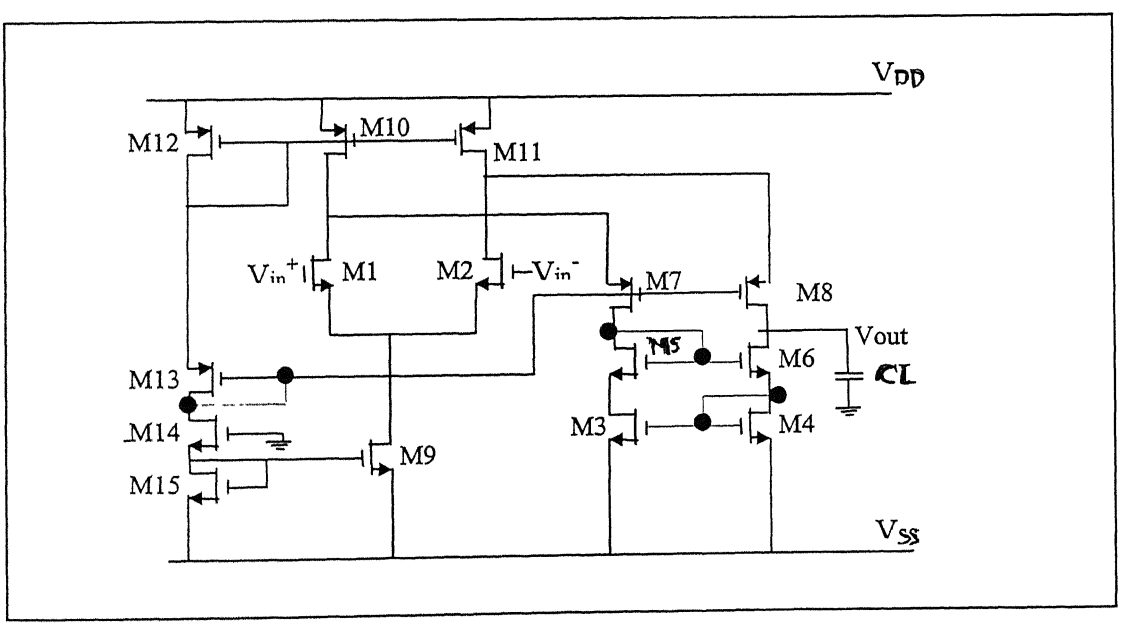


Fig: 3.7. Transistor level schematic of OP-Amp.

As shown in Fig:3.7 after obtaining complete architecture/circuit of OP-Amp in terms of sub-components like MOS Transistors, we need to verify the functionality and the performance of the chosen circuit through simulation of a structural representation of the whole circuit of OP-Amp. If the results obtained do not match with the behavioral level simulations, then appropriate modifications, in the chosen architecture is done and simulation is repeated. This architectural description of OP-Amp is done using analog Hardware Description Language or SPICE.

☞

The series of steps involved in the transformation of DAC from behavioral level to transistor level are repeated for comparator and latch also. Once we obtain a complete transistor level circuit of the ADC then we need to perform the complete system simulation to evaluate the system performances. But performing the whole system simulation of ADC at this transistor level is very time consuming because of large no of transistors and usage of complex models for these transistors. So there is a need to develop some abstract level model of complete ADC system with inclusion of the parameters that affect the performance of the system, so that the whole system simulation can be completed in lower run times. The system simulation for digital systems can be done in more abstract manner at distinct levels like switch level, gate level, RTL level and behavioral level. Unlike Digital Systems, there are no specific distinct levels for abstraction in Analog/Mixed Signal systems. Here we have done the system level simulation by using abstraction at one level higher than OP-Amps, comparators and transistor switches.

3.3 System level simulation using abstraction

As mentioned above, there are no well defined universally applicable abstraction levels in the analog domain. The level of abstraction also depends on the class of circuits and within a class on the purpose of abstraction. In the present case, here objective is to obtain system level characteristics such as Differential Non-Linearity (DNL) and Integral Non-Linearity (INL) in the Analog-to-Digital Converter through simulation at a suitable abstraction level. The choice of abstraction level has to balance the conflicting requirements of accuracy and speed of simulation. A very high abstraction level such as

the ' architectural-level ' would result in very fault simulation times but it would be very difficult to estimate the performance accurately. With these requirements in mind, an abstraction level that replaced all OP-Amps, transistors and capacitors by a behavioral model was chosen.

The parameters Differential Non-Linearity (DNL) and Integral Non-Linearity (INL) errors, are the most important specifications of ADC. Differential Non-Linearity (DNL) is the difference between an actual step width (for Analog-to-Digital Converter) or step height (for Digital-to-Analog Converter) and the ideal value of 1LSB (Least Significant Bit). Integral Non-Linearity (INL) is the deviation of the values on the actual transfer function from a straight line. The straight line is a line drawn between end points of a transfer function. The name integral non-linearity derives from the fact that the summation of different non-linearities from the bottom-up to a particular step determines the value of integral non-linearity at that step. The deviations are measured at the transitions from one step to the next for Analog-to-Digital Converter and for Digital-to-Analog Converter they are measured at each step. The factors that affect the DNL and INL are finite gain, offset voltage, mismatches of capacitance and clock feed-through due to overlapping capacitance.

The abstraction level chosen to determine these characteristics must be in such a way that it reflects the effects of these parameters accurately. The nature of the switched capacitor circuit changes with each clock cycle. There is a distinct arrangement of OP-Amps, transistors and capacitors for each cycle. This structural representation is converted to a behavioral form (analytical equations) along with effects of gain, offset voltage, capacitance mismatch and overlapping capacitance. This methodology is described in detail for the ADC in the next section.

3.3.1 Abstraction methodology for ADC

In this abstraction methodology, the digital-to-analog converter (DAC) and latch are described at behavioral abstraction level, while the remaining circuitry which consists of transistors, capacitors and OP-Amps are considered as a single block and described

behaviorally. The behavioral abstraction methodology used for DAC is explained in detail in the next section. The behavioral description of latch is simple and straightforward, as it doesn't incorporate any parameters that affects the overall system performance. The remaining circuitry that consists of OP-Amps, transistors and capacitors reduces to two different structures depending upon the state of sample(S) signal and Φ_1 signal. Each of this structural representation of the block is converted to behavioral form by describing it in the form of analytical equations as shown below.

If the sample signal is in high state, hold signal in low state i.e. S=1,H=0 and Φ_1=1, Φ_2=0 and Φ_3=0, then the remaining circuitry apart from DAC and latch of analog-to-digital converter (in Fig:3.3) forms the following structure shown in Fig:3.8.

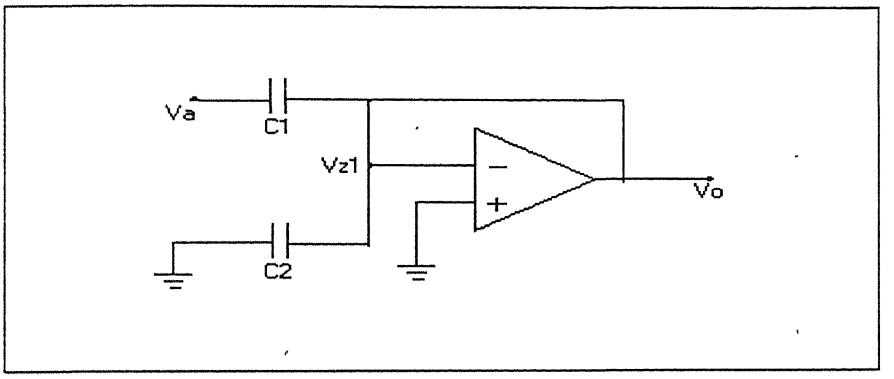


Fig:3.8. Equivalent circuit of ADC when S=1,H=0 and Φ_1=1.

The input analog voltage is stored in capacitor C_1. From simple calculations the voltage V_{z1}, voltage across capacitors C_1,C_2 and output voltage V_o are obtained as shown in eq-(3.8) and eq-(3.9).

$$Vo = V_{c2} = Vz1 = [(-AV_{off})/(1+A)] \quad (3.8)$$

$$V_{c1} = [(-AV_{off})/(1+A)] - [V_a] \quad (3.9)$$

When either of the signal Φ_1 or S goes low (i.e Φ_1=0 or S=0) the block of ADC (in Fig:3.3) apart from DAC and latch reduces to the following structural representation

shown in Fig:3.9. This structure remains same even when the state of signals Φ_2 and Φ_3 changes.

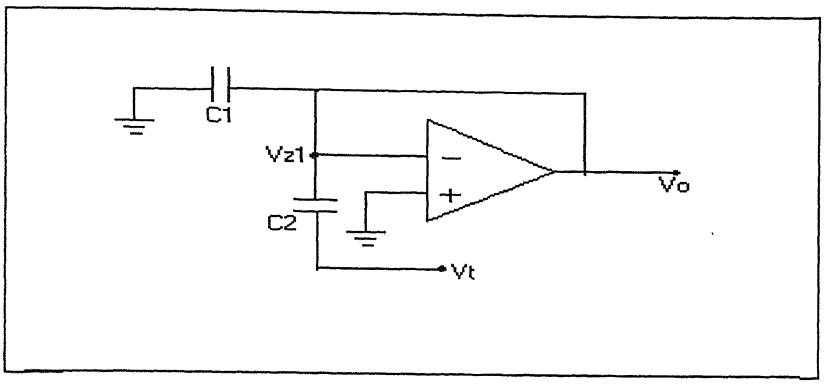


Fig:3.9. Equivalent Circuit of ADC when S=0 or Φ_1=0

If the voltage V_{z1} is greater than zero then output will be V_{sat} else it would be $-V_{sat}$. The voltage V_{z1} with simple analysis can be calculated and is given by the following eq-(3.10)

$$V_{z1} = [-(AV_{off})/(1+A)] + [(-C_1V_a + C_2V_t)/(C_1+C_2)] + V_{off} \qquad (3.10)$$

The output voltage V_o is $+V_{sat}$ or $-V_{sat}$ depending upon the value of V_{z1} is greater or lesser than zero, as this is a comparator stage.

3.3.2 Behavioral level Abstraction for DAC:

The complete architecture of DAC shown in Fig:3.5. is reduced to different forms with change in states of Sample (S), hold (H), Φ_1 ,Φ_2 and Φ_3 signals and also depends upon the previous bit value b_I . Here the abstraction is done by considering the whole DAC as a block, which reduces to different structural representations with change in states. Each structural representation consisting of OP-Amps, transistors and capacitors are described behaviorally as a single block using analytical equations, with inclusion of parameters like finite gain, offset voltage, capacitance mismatch and overlapping capacitance as shown below.

S=1,H=0,Ø1=1,Ø2=0,Ø3=0

When the sample signal (S='1') is high (i.e. hold signal is low) the equivalent circuit of digital-to-analog converter (DAC) has three forms depending upon the state of signals Φ_1 ,Φ_2 and Φ_3. This circuit is independent of the state of the previous bit value, and this is true only when sample signal is high (S='1') and hold signal is low (H='0'). Let's consider the first state i.e. when Φ_1 = '1', Φ_2 = '0' and Φ_3 = '0', then the complete architecture of digital-to-analog converter shown in Fig:3.5 is reduced to the circuit shown in Fig:3.10.

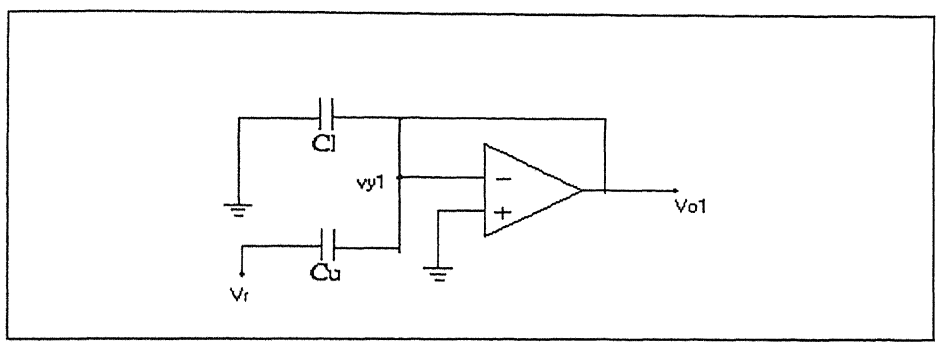


Fig:3.10. Equivalent circuit of DAC at S=1,H=0,Φ_1=0, Φ_2=1 and Φ_3=0.

The voltage V_{y1}, voltage across capacitors C_l, C_u and voltage V_{o1} are obtained with simple analysis and they attain the values as shown in eq-(3.11) and eq-(3.12).

$$V_{y1} = V_{cl} = V_{o1} = (-AV_{off})/(1+A) \quad (3.11)$$

$$V_{cu} = [(-AV_{off})/(1+A)] - V_r \quad (3.12)$$

S=1,H=0,Ø1=0,Ø2=1,Ø3=0

When the next state Φ_1 = '0', Φ_2 = '1' and Φ_3 = '0' reaches, then the digital-to-analog converter shown in Fig:3.5 is reduced to the circuit shown in Fig:3.11. In this

state voltage across capacitor C_t rises to the voltage stored in capacitor C_u in previous state.

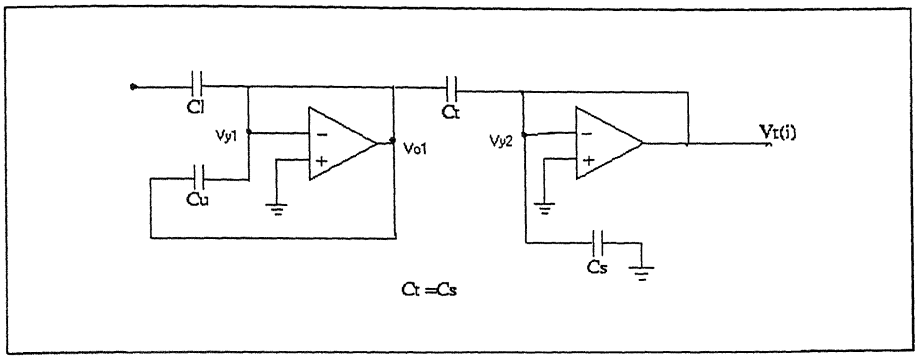


Fig:3.11. Equivalent circuit of DAC at S=1,H=0,Φ_1=0, ,Φ_2=1 and Φ_3=0.

The voltages V_{y1},V_{o1} and V_{y2} are obtained with simple analysis and are given by eq-3.13 to eq-3.15. The voltage across capacitor C_l remains same, but voltage across capacitors C_u and C_t changes and are given according to the eq-(3.16) and eq-(3.17). In this state the voltage across C_u is transferred to C_t.

$$V_{y1} = [\ \{\ (-AV_{off})/(1 + A)\ \} - V_r - AV_{off}\ ][1/(1+A)] \quad (3.13)$$

$$V_{o1} = [(A^2\ V_{off})/(1+A)^2] + [(AV_r)/(1+A)] - [(AV_{off})/(1+A)] \quad (3.14)$$

$$V_{y2} = V_{cs} = V_t = (-AV_{off})/(1+A) \quad (3.15)$$

$$V_{cu} = [\ (-AV_{off})/(1+A)\ ] - V_r \quad (3.16)$$

$$V_{ct} = [\ (A^2\ V_{off})/(1+A)^2\ ] - [\ (AV_r)/(1+A)\ ] \quad (3.17)$$

From the equations in the previous state and present state we can observe the voltage across the capacitor C_u has a relation with the previous value of voltage across capacitor C_u as given in eq-3.18.

$$V_{ct} = [(AV_{cu}')/(1+A)] \quad (3.18)$$

Where V_{cu}' is value of V_{cu} in the previous state.

S=1,H=0,Ø1=0,Ø2=0,Ø3=1

When the state Φ_1 = '0', Φ_2 = '0' and Φ_3 = '1' reaches, then the digital-to-analog converter shown in Fig:3.5 is reduced to the circuit shown in Fig:3.12

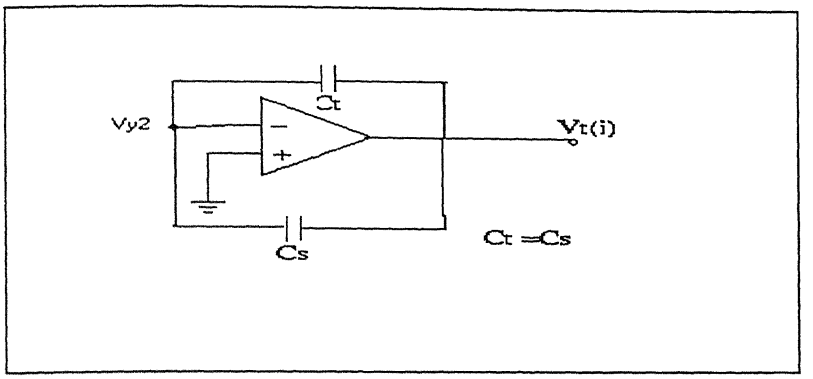


Fig:3.12. Equivalent circuit of DAC at S=1,H=0,Φ_1=0, ,Φ_2=0 and Φ_3=1.

This circuit in Fig:3.12 is a divide by two circuit and hence a divide by two operation occurs between the voltages across C_t and C_s. Here in this state the final voltages across capacitors C_t and C_s are equal as given in eq-(3.19). The output voltage V_t is given by eq-(3.20).

$$V_{ct} = V_{cs} = [\ \{-(C_tA^2\ V_{off})/(1+A)^2\} - \{C_tAV_r/(1+A)\} - \{C_sAV_{off}/(1+A)\}\ ][\ 1/(C_t + C_s)\ ] \quad (3.19)$$

$$V_t = [\ \{(C_tA^2\ V_{off})/(1+A)^2\} + \{C_tAV_r/(1+A)\} + \{C_sAV_{off}/(1+A)\}\ ][A/(1+A)][\ 1/(C_t + C_s)\ ] - [AV_{off}/(1+A)] \quad (3.20)$$

From the equations in the previous state and equations in this state the voltage across capacitor C_t can be computed easily using eq-(3.21).We can also show that the value of V_t can be computed using the following equatiion (3.22).

$$V_{ct} = [(C_tV_{ct}' + C_sV_{cs}')/(C_t + C_s)] \quad (3.21)$$

$$V_t = [-V_{ct}A/(1+A)] - [\ (AV_{off})/(1+A)\ ] \quad (3.22)$$

Similarly the analysis is done for remaining states of the conversion cycle which are given briefly in appendix-III.

Here we have also simulated the DAC block in SPICE and compared those results with that obtained using our abstraction methodology. Fig:3.13 and Fig:3.14 shows the SPICE simulation results and VHDL simulation results for ideal 3-bit DAC (i.e gain is very large, and no capacitance mismatches) for input code of '000'. The period of sample signal 'S' is taken 360ms. The plots clearly shows that it matches quantitatively, though the timing has not matched accurately as because we have considered the analysis of ADC at discrete stages in VHDL, where continuous modeling cannot be achieved.

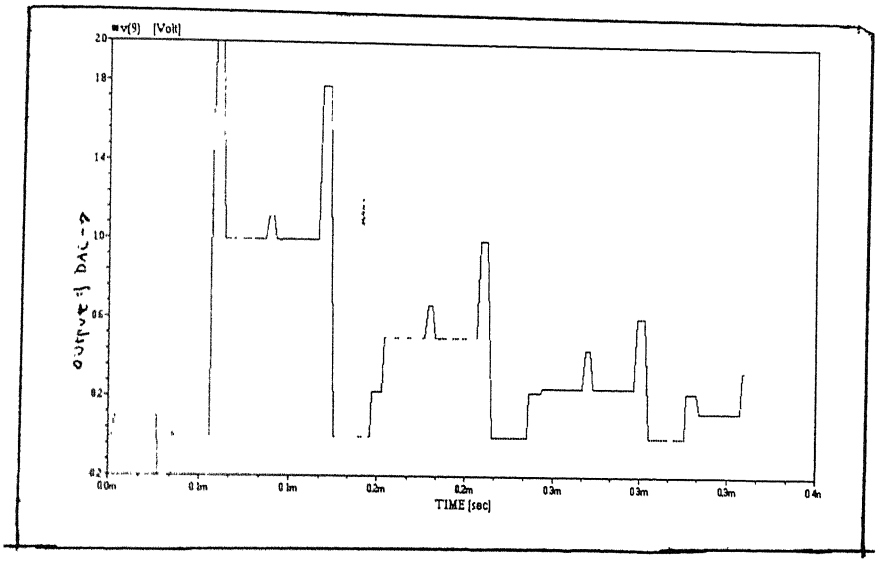


Fig:3.13. SPICE simulation results for 3-bit ADC at high gain

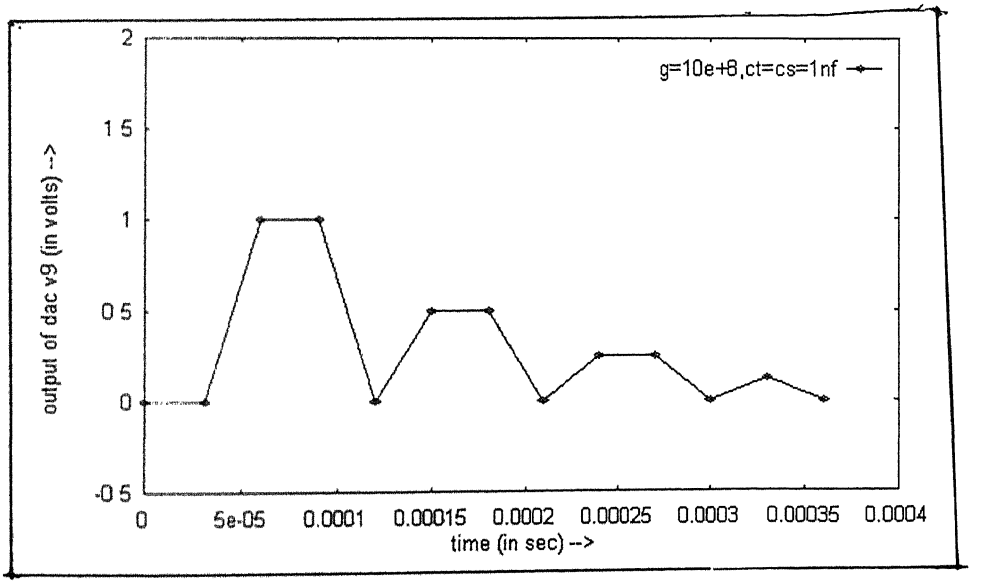


Fig:3.14. VHDL simulation results for 3-bit ADC at high gain

Similarly Fig:3.15 and Fig:3.16 shows the simulation results obtained for DAC with finite gain and finite capacitance mismatch from SPICE and VHDL respectively. The period of sample 'S' signal is kept same, and we have obtained the exact quantitative values. This shows that the analysis work that we have done in this abstraction methodology is perfect.

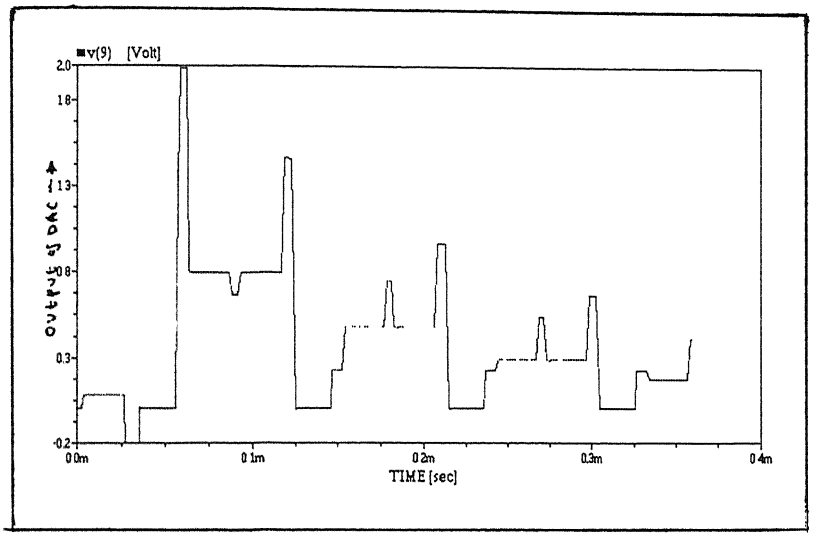


Fig:3.15. SPICE simulation results for 3-bit ADC at low gain and mismatch of capacitance

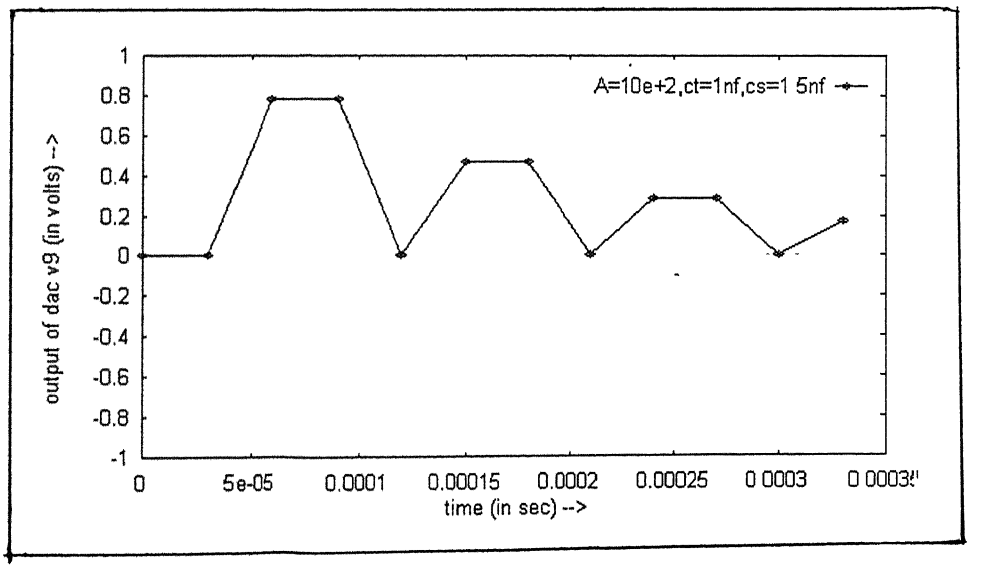


Fig:3.16. VHDL simulation results for 3-bit ADC at low gain and mismatch of capacitance

3.4 *Non-ideal parameters and its effect on system simulation*

Here we have considered the non-ideal parameters like finite gain (A), offset voltage (V_{off}), capacitance mismatch and overlapping capacitance's and performed overall system simulation to study the effects on the overall circuit by calculating differential non-linearity (DNL) and integral non-linearity (INL) errors.

3.4.1 *Effect of Offset Voltage V_{off}*:

Ideally the offset voltage of an OP-Amp is zero, but practically there exists a finite amount of voltage, which affects the circuits severely especially in the comparator blocks. In this circuit the Offset Voltage doesn't effect the performance of the circuit. This is the most widely used method to compensate the effect of the offset voltage. For e.g. consider the first block of ADC in Fig:3.3. which is shown in Fig:3.17.

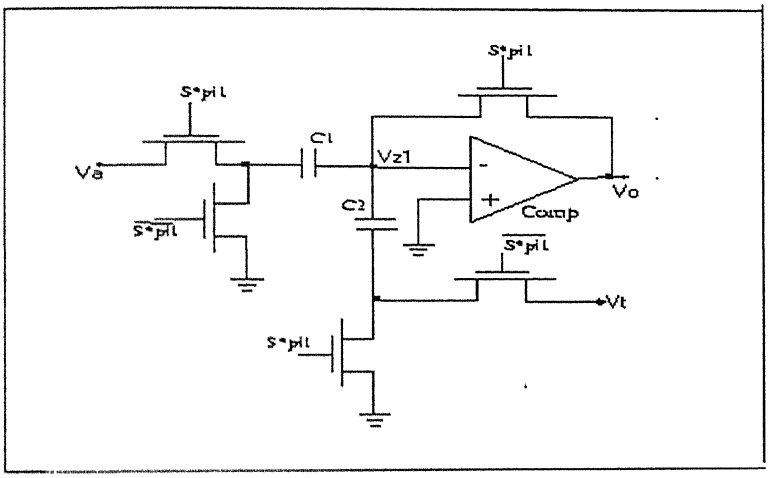


Fig:3.17 A block of Analog-to-Digital converter

Initially before conversion starts i.e. when S=1,H=0, Φ_1=1,Φ_2=0 and Φ_3=0 then the offset voltage is stored in both the capacitors C_1 and C_2 as given below:

$$V_{c1} = [(-AV_{off})/(1+A)] - V_a \quad (3.23)$$

$$V_{c2} = [(-AV_{off})/(1+A)] \quad (3.24)$$

In the subsequent next state these capacitor voltages cancel the effect of offset voltage, provided the gain is very high and the value of V_{y1} is given below:

$$V_{y1}= [-(AV_{off})/(1+A)] + [(-C_1V_a + C_2V_t)/(C_1+C_2)] + V_{off} \quad (3.25)$$

This equation reduces to the ideal equation $V_{y1}= [(-C_1V_a + C_2V_t)/(C_1+C_2)]$ if gain is very high, thus indicating that the circuit is independent of offset voltage V_{off}. Similarly at all stages the offset voltage V_{os} is stored in the capacitors in the first state and these capacitor voltages practically cancel the offset voltage of OP-Amp in the subsequent conversion steps if the OP-Amp has a reasonably large gain, thereby assuring offset-free operation. The DAC transfer characteristics do not change even with change in V_{ioff}. This is shown in Fig: 3.18.

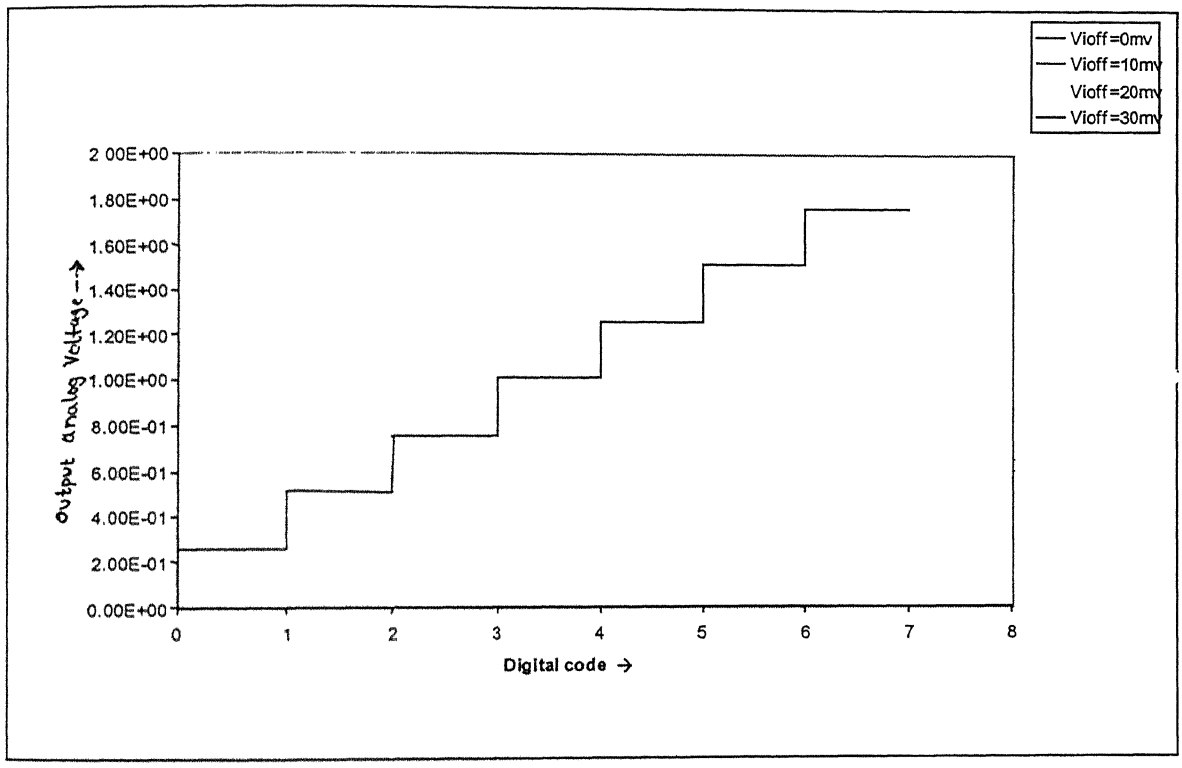


Fig:3.18 Transfer Characteristics of 3-bit DAC with variation in offset voltage at fixed gain of 10,000

3.4.2 *Effect of Capacitance Mismatches*

Practically there exists capacitance mismatches between capacitors C_1, C_2 and C_t, C_s in ADC circuit shown in Fig:3.3. the effect of mismatch between C_1, C_2 on output characteristics of ADC is illustrated here. For example consider the block shown in Fig:3.19 which is a part of ADC circuit in Fig:3.3. Here the mismatch between capacitors C_1 and C_2 are taken into account.

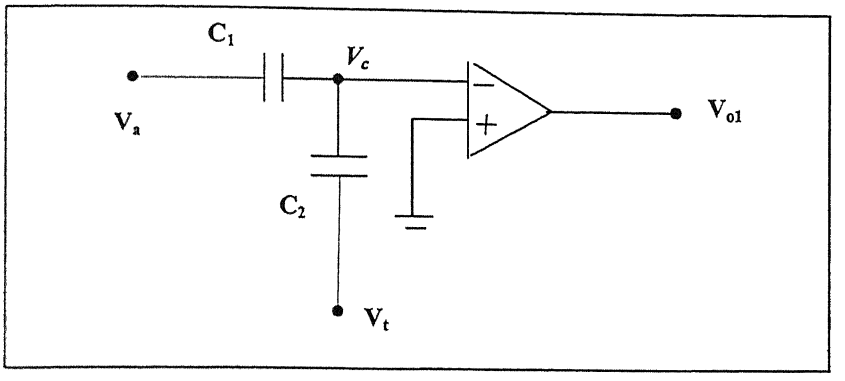


Fig:3.19 Capacitance mismatch between C1 and C2 at the input stage of ADC

From simple analysis the value of V_c is obtained as given below.

$$V_c = (- V_a C_1 + C_2 V_t)/ (C_1 + C_2) \qquad (3.26)$$

In the ideal case [$C_1 = C_2 = C$] eq-3.26 reduces to the following equation shown below.

$$V_c = (- V_a + V_t)/ 2 \qquad (3.27)$$

Taking $C1 = C + \delta C_1$ and $C_2 = C + \delta C_2$ we obtain eq-3.28.

$$V_c' = (- V_a (C + \delta C_1) + (C + \delta C_2) V_t)/ (2C + \delta C_1 + \delta C_2) \qquad (3.28)$$

The error voltage given by $\Delta V_c = V_c' - V_c$ cab be written as shown in eq-3.29.

$$\Delta V_c = \frac{[\frac{(\Delta C)}{2}][\frac{(V_a + V_t)}{2}]}{C + [\frac{(\Delta C)}{2}]} \qquad (3.29)$$

Where $\Delta C = \delta C_2 - \delta C_1$.

This effect of mismatch of capacitance's between C_1,C_2 and C_t,C_s on output characteristics of ADC and DAC is taken into account and those variation of the output

characteristics with increase in mismatch of capacitance's are shown in Fig:3.20 and Fig:3.21 respectively.

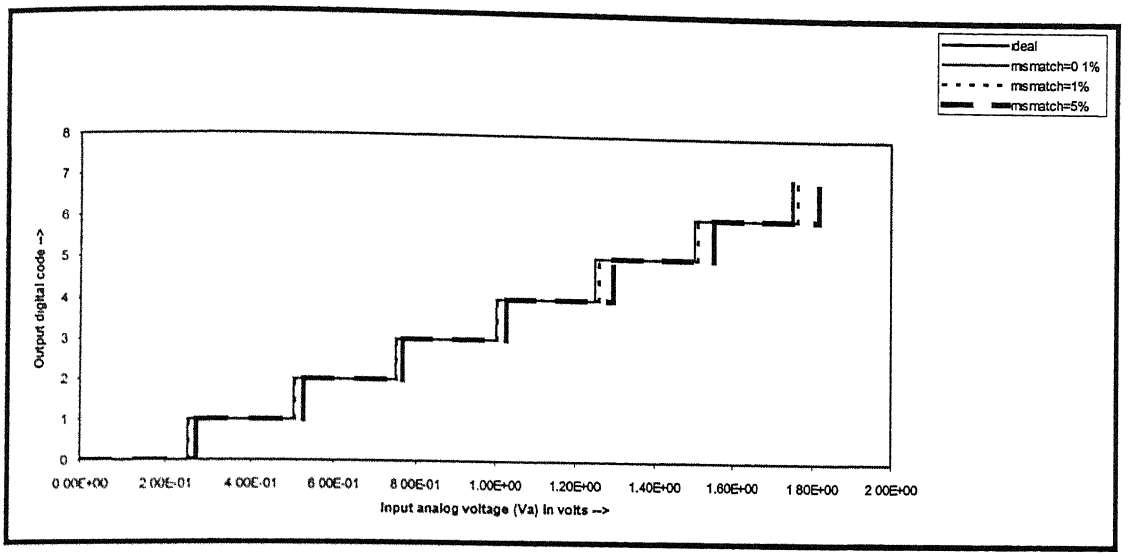


Fig:3.20. The variation of transfer characteristics of 3-bit ADC for varying mismatch of capacitance with fixed gain of 10,000.

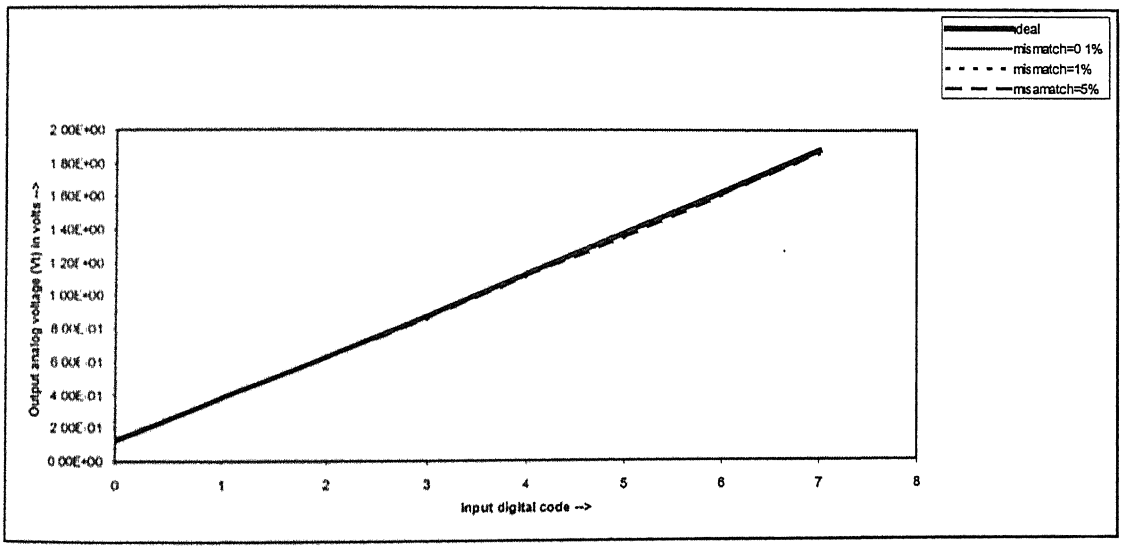


Fig:3.21. The variation of transfer characteristics of 3-bit DAC for varying mismatch of capacitance with fixed gain of 10,000.

3.4.3 *Effect Of Finite Gain (A)*

Ideally OP-Amp's gain is infinite, but practically due to its finite value it effects most of the circuits. Here if the gain is large it also neutralizes the effect of offset voltage. High OP-Amp gain is needed to ensure complete charge transfer between the capacitors. As gain reduces it affects the output characteristics of analog-to-digital converter and digital-to-analog converter as shown in Fig:3.22 and Fig:3.23.

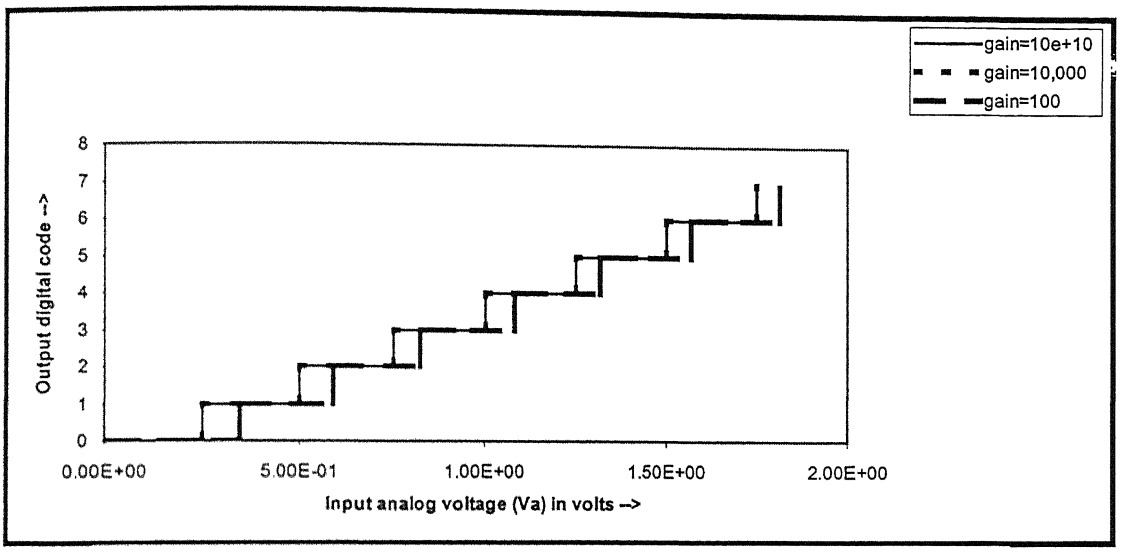


Fig:3.22. The transfer characteristics of 3-bit ADC for varying gain.

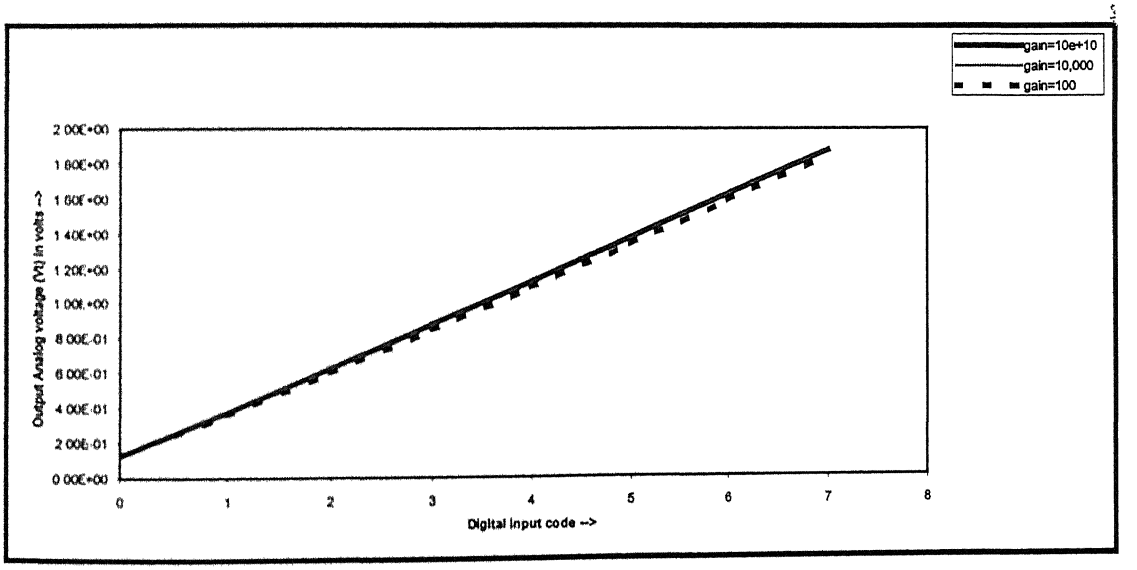


Fig:3.23. The transfer characteristics of 3-bit DAC for varying gain.

3.4.4 *Effect of Overlapping Capacitance*

When a MOS transistor switch is in off condition then the overlapping capacitance that is present between gate-drain and between gate-source inject charge into the surrounding circuit nodes and thus voltage at those points deviate from their normal values thus leading to clock feed through effect. This effect does not come into picture when the MOS transistor is in on state.

This overlapping capacitance effect is taken into account in the analog-to-digital converter circuit. We can observe from Fig:3.3 that the clock-feed-through effect appears at MOS transistor M_4. Consider the figure shown in Fig:3.20 which is obtained from ADC circuit in Fig:3.3. The overlapping capacitance's C_{ol1} and C_{ol2} are present practically as shown in Fig:3.24.

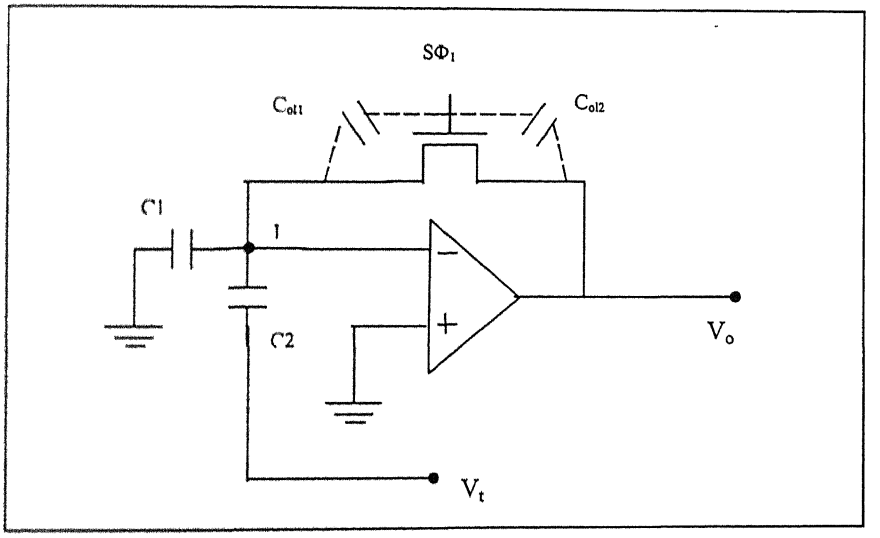


Fig:3.24 ADC with overlapping capacitance's.

From the charge conservation at node (1) we can obtain the error voltage due to clock feed through effect and is given by eq-(3.31)

$$V_{eft} = [C_{ol1}/(C_1+C_2+C_{ol1})][\ \{AV_{off})/(1+A)\} + V_{cc} + \{(C_1V_a-C_2V_t)/(C_1+C_2)\}] \qquad (3.31)$$

The effect of this overlapping capacitance's dominates with increase in the capacitance value. The variation of output characteristics of Analog-to-Digital Converter is shown in Fig:3.25

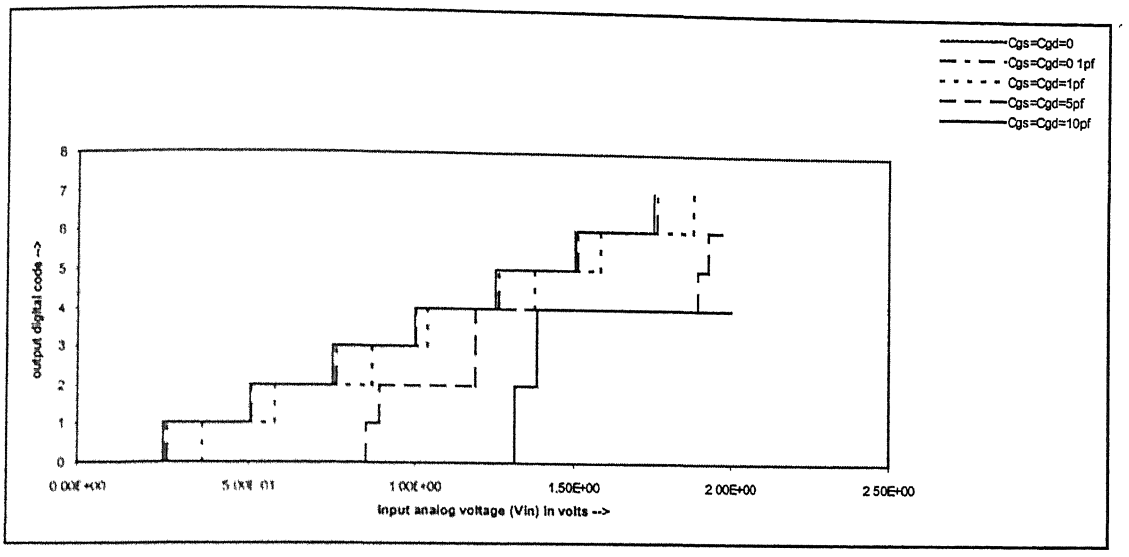


Fig: 3.25. Transfer characteristics of 3-bit ADC with varying value of overlapping capacitance

From Fig:3.25 we can say that with increase in overlapping capacitance the output characteristics of Analog-to-Digital Converter (ADC) deviates to much extent from the ideal characteristics of ADC.

The Differential Non-Linearity (DNL) and Integral Non-Linearity (INL) are obtained from the ideal and non-ideal characteristics of Analog-to-Digital Converter. For example the DNL and INL for ADC at gain of 10,000, capacitance mismatch of 1% and overlapping capacitance of 0.1pf are shown in Fig:3.26 and Fig:3.27, and similarly the DNL and INL are obtained for gain of 1000, capacitance mismatch of 5% and overlapping capacitance of 5pf as shown in Fig:3.28 and Fig:3.29. From Fig:3.26 and Fig:3.27 we can say that the no code would be missed during the operation as maximum DNL is 0.5LSB, and maximum INL is 0.5LSB which are less than 1LSB, hence these parameter values do not effect the performance of the system seriously. Now consider the Fig:3.28 and Fig:3.29 the maximum DNL is 1.5LSB which is more than 1LSB, hence here code7 is missed during the operation of the circuit. These results indicate that INL is

more effected due to finite gain and is maximum when all bits are zero. The results are shown

below.

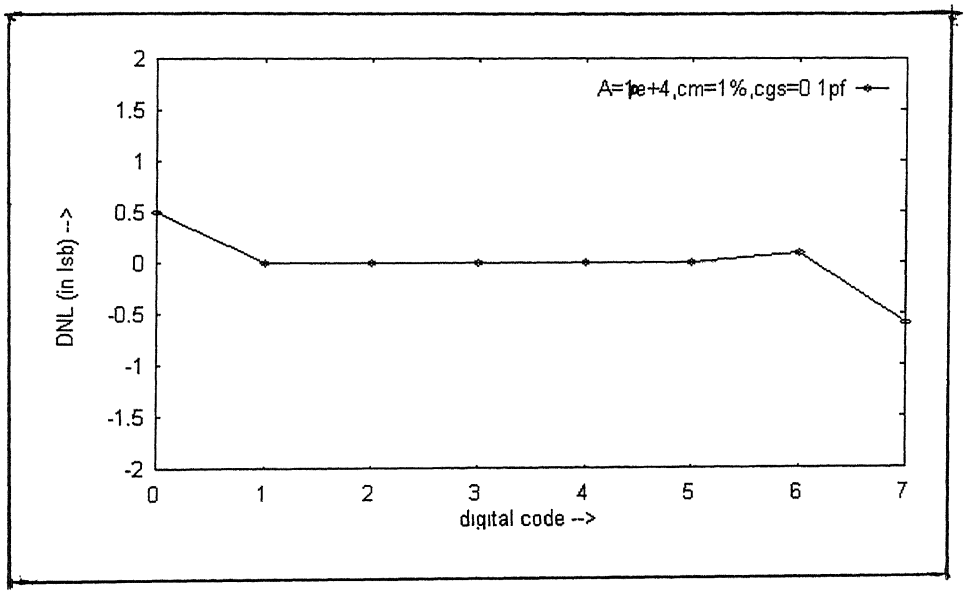


Fig: 3.26. Differential Non-Linearity (DNL) for ADC with A=10e+4,cm=1%,Cgs=0.1pf

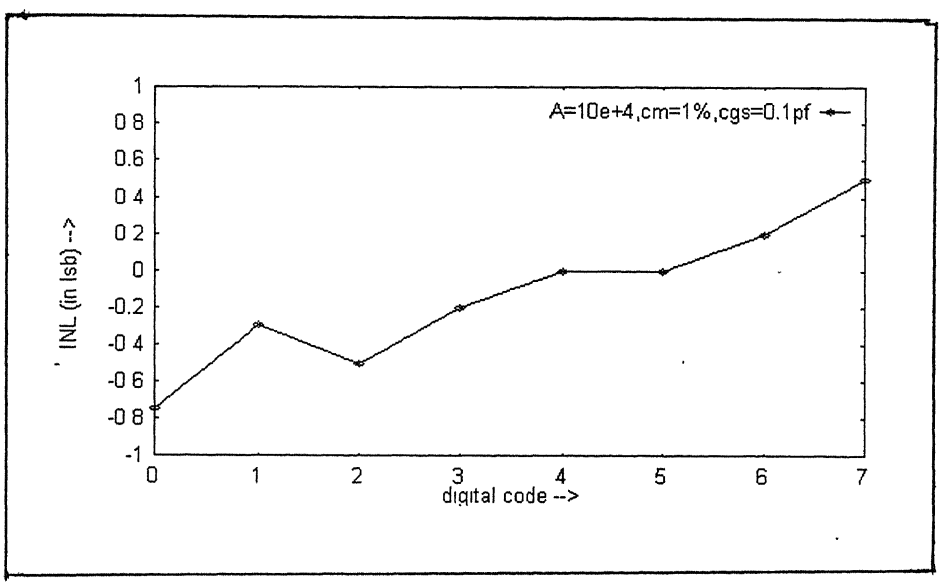


Fig: 3.27. Integral Non-Linearity (INL) for ADC with A=10e+4,cm=1%,Cgs=0.1pf

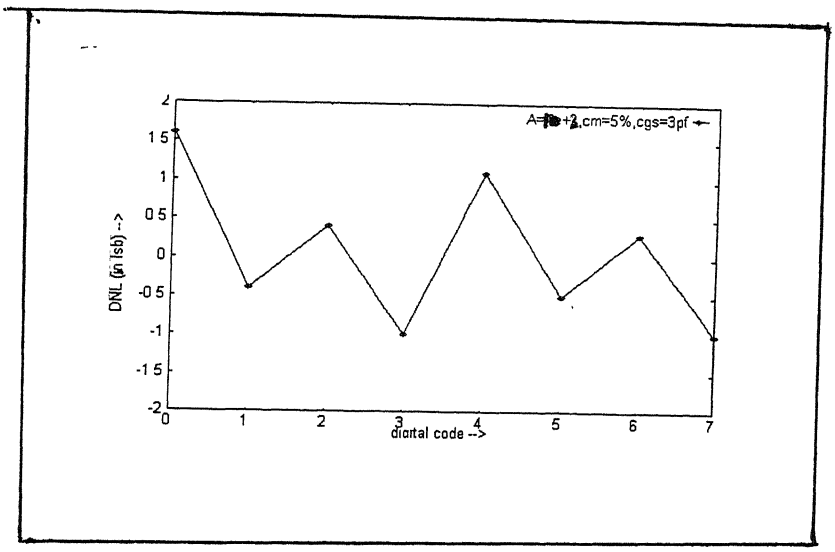


Fig: 3.28. Differential Non-Linearity (DNL) for ADC with A=10e+3,cm=5%,Cgs=5pf

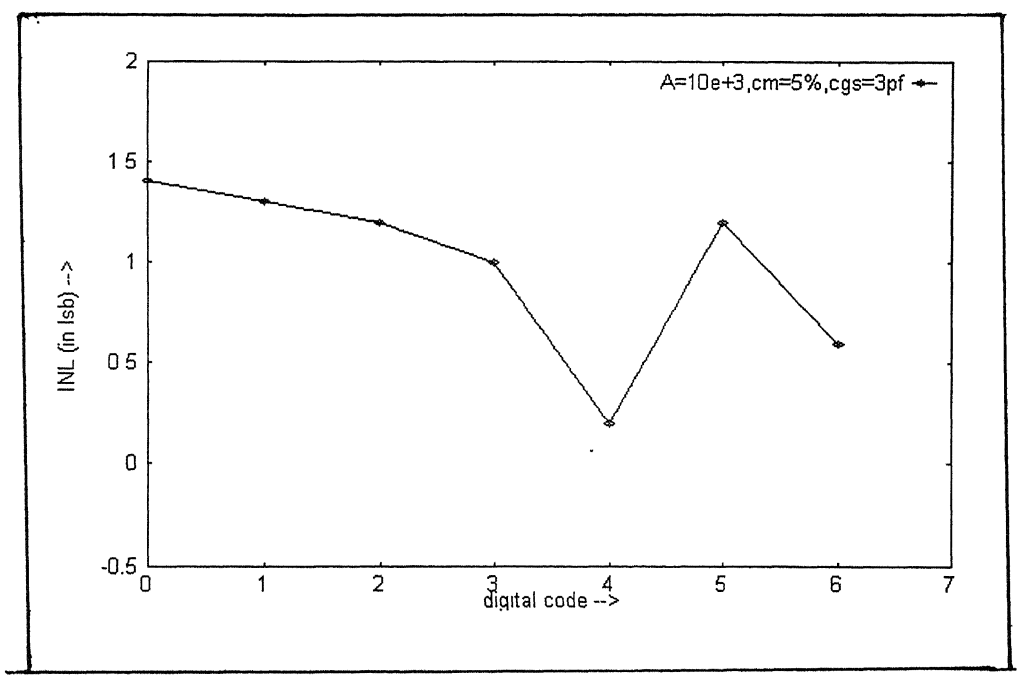


Fig: 3.29. Integral Non-Linearity (INL) for ADC with A=10e+3,cm=5%,Cgs=3pf

CHAPTER-4

CONCLUSION

In the top-down design methodology, the overall characteristics of the design become increasingly better defined as the design migrates to lower levels. The best estimate of the quality of the design that can be obtained prior to the actual fabrication and testing is through the simulation of a transistor level schematic of the design (before and after layout design) . This however becomes increasingly difficult as circuit complexity increases. To overcome these problems, the use of well known techniques for coping with complexity namely hierarchy and abstraction in conjunction with versatile modeling features of a hardware description languages were described for easing system level simulation. This was illustrated through the example of a switched capacitor A/D converter modeled in VHDL. An abstraction level that substitutes groups of opamps, transistor switches and capacitors by a simple behavioral description including effects of parameters such as gain, offset voltage, capacitance mismatch, overlapping capacitance effects was developed to estimate the differential and integral non-linearity chracteristics

of the final design. The use of complex ADC simulation at these abstraction level for deciding suitable values for gain of opamp and capacitance mismatch have been shown

The abstraction methodology that we adopted here in this work can be used for wide range of switched capacitor circuits. In future, there is a need to develop a standard abstraction methodology for analog/mixed signal systems, which can be accepted universally similar to that used for digital devices.

BIBILOGRAPHY

[1] Ping Waili, Michael. J. Chin, Paul. R. Gray and Rinaldo Castello, "A Ratio–independent algorithmic analog-to-digital conversion technique," *IEEE Journal of Solid - State Circuits*, Vol-19, No-6, p-828, Dec 1984.

[2] Ricardo. E. Suarez, Paul. R. Gray and David. A. Hodges, "All-MOS charge redistribution analog-to-digital conversion," *IEEE Journal of Solid - State Circuits*, Vol-10, No-6, p-379, Dec 1975.

[3] Hiroki Matsumoto and Kenzo Watanabe, "Switched Capacitor algorithmic Digital-to-Analog Converters," *IEEE Trans. on Circuits and Systems*, Vol-33, No-7, p-721, July 1986.

[4] Hans Sahm, Claus Mayer, Jorg Pleickhardt, Johannes Schock and Stefan Spath, "VHDL Development System and coding standard," *IEEE Design and Test of Computers*, p-777, 1997.

[5] Roper Lipsett, Erich Manschner and Moe Shahdad, "VHDL-The Language," *IEEE Design and Test of Computers*, p-28, April1986.

[6] Satomi Ogawa and Kenzo Watanabe, "A Switched Capacitor Successive-Approximation Analog-to-Digital converter," *IEEE Trans. on Instrumentation and Measurement*, Vol-42, No-4, p-847, August 1993.

[7] J. H. Aylor, R. Waxman and C. Scarratt, "VHDL-Feature Description and Analysis," *IEEE Design and Test of Computers*, p-17, April 1986.

[8] Jerzy Dabrowski and Andrzej Pulka, "Discrete Approach to PWL Analog Modeling in VHDL environment," *Analog Integrated Circuits and Signal Processing*, p-3, June 1997.

[9] G. Ruan, J. Vlach, J. Barby and A. Opal, "Analog Functional Simulator fc_ multilevel systems," *IEEE Transactions on CAD*, Vol-10, No-5, p-565, May 1991.

[10] A. J. Perkins, M. Zwolinski, C. D. Chalk and B. R. Wilkins, "Fault Modeling and Simulation using VHDL-AMS," *Analog Integrated Circuits and Signal Processing,* p-53, June 1997.

[11] Lunye and Harold. W. Carter, " Analog Behavior Modeling and Processing using Ana VHDL," *Analog Integrated Circuits and Signal Processing*, p-85, Nov 1996.

[12] Alain Vachoux, " Analog and Mixed Signal extensions to VHDL," *Analog Integrated Circuits and Signal Processing*, pp-13, June 1998.

[13] Yaohan Chu, Donald. L. Dietmayer, Fredrick. J. Hill, Mario. R. Barbacci, Charles. W. Rose and Bill Johnson, "Three Decades of HDL's," *IEEE Design and Test of Computers*, p-69, June 1992.

[14] J.Dabrowski, "Functional-level analogue macromodeling with piece wise linear signals," *IEE Proc.-Circuits Devices Syst.*, Vol-146, No-2, p-77, April 1999.

[15] Charles H. Small, "Mixed-Signal HDL's unite two worlds," *Computer Design*, p-44, October 1997.

[16] Farid Najm, "Simulation and Modeling-AHD Languages-A Must for Time-Critical Designs," *Circuits and Devices*, p-12, July 1996.

[17] Michael Depeyrot, "DSP and VHDL-AMS should be taught jointly," *Computer Design*, p-85, October 1998.

[18] R.Gregorian and G.C.Temes, "Analog MOS Integrated Circuits for Signal Processing",*Wiley,New York*,1986.

[19] P.R.Gray and R.G.Meyer, "Analysis and Design of Analog Integrated Circuits", *John Wiley & Sons, Inc.*, 1993.

[20] Mohammed Ismail and Terri Fiez, "Analog VLSI-Signal and Information Processing", *McGraw-Hill International Editions*, 1994.

[21] Steven S. Leung and Michael A. Shanbaltt, "ASIC System Design with VHDL: A Paradigm", *Kluwer Academic Publishers*, 1991.

[22] Stefan Sjoholm and Lennart Lindh, " VHDL for Designers", *Prentice Hall*, 1997.

[23] J. Bhasker, " VHSIC Hardware Description Language", 1994.

[24] Muhammad H. Rashid, "SPICE for Circuits and Electronics using PSPICE", *Prentice Hall of India Pvt. Ltd.*, 1995.

[25] R. Jacob Baker, Harry W. Li and David E. Boyce, "CMOS circuit design, Layout, and Simulation", *IEEE Press Series on Microelectronic Systems*,1998.

APPENDIX-I

Here a brief history of the evolvement of VHDL is given and then summarized it's important features that made the language put into use. VHDL stands for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. This is the most widely used and industry accepted standard language. VHDL provides a standard textual means of description for hardware components at levels ranging from logic gate level to structural level.

HISTORY OF VHDL:

Each vendor has developed his own Hardware Description Language (HDL) for design, documentation and simulation purpose. There arised many problems among the vendors in exchanging their ideas in the designing and thus it was not portable. Then during that period the U.S Defense Authority Research Program Academy (DARPA) has given a project to a team comprising of IBM, TEXAS and INTERMETRICS for developing a standard Hardware Description Language. Then as a result of that project VHDL (Very high speed IC Hardware Description Language) was actually developed in the year 1983. It later became as an IEEE standard language in the year 1987. This is the first version of VHDL, then second version that corrects language inconstancies and includes new mechanisms has been released in the year 1993. Major developments have also been taken into account in the year 1993 & 1998.

The design of a system using this VHDL language is technology independent, which is an added advantage as technology, is rapidly changing day by day. This VHDL has almost all features that are existing in different languages that exist today and past. VHDL has gained considerable importance as it is today being supported in all major Electronics Design Automation environments available in the market. It is also going beyond its original goals as it also efficiently supports automatic synthesis, formal proof and testing. As such VHDL is acting as a unified medium that supports a large part of the design process, from the abstract specifications to the logical implementation into gates.

FEATURES OF VHDL:

VHSIC Hardware Description Language (VHDL) is a vital language among all the Hardware Description Languages existing today. It is a unified medium language that supports a large part of the design process, from the abstract specifications to the logical implementation into gates. The VHDL supports Mixed level and Mixed mode descriptions. Mixed level implies using elements at different abstraction levels. Mixed mode implies using both combinational and sequential descriptions in the same model. Let's discuss the features of this language when it was IEEE standardized. The features of VHDL are described as below:

- **DESIGN ENTITY CONCEPT**: VHDL provides well-considered mechanism supporting alternative design representations unlike other Hardware Description Languages (HDL's) which are neglecting the factor that the design can be described at many levels. A design entity models hardware of any complexity: for eg: a design entity may model a logic gate, a flip-flop, a control unit, or a computer system.

  A design entity is composed of an interface and one or more alternative bodies as shown in the figure Fig:AI-1.

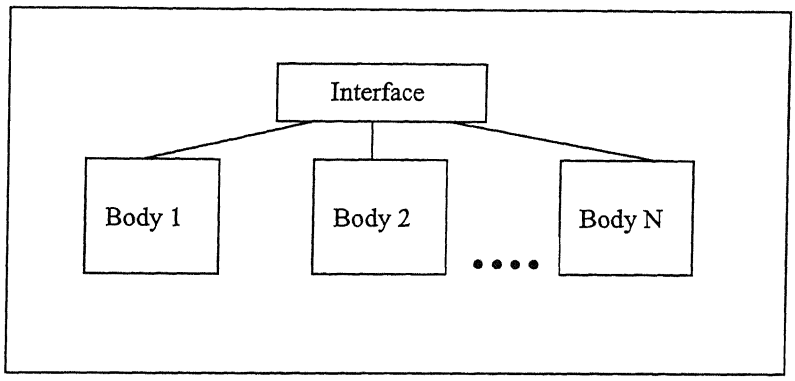


Fig :AI-1 A Design Entity

The interface contains a set of definitions common to alternative bodies. Such definitions capture the hardware entity's external view and specify communication channels between the design entity and the outside world. Each alternative body describes the alternate view of the hardware entity. For eg: one body may describe a hardware entity's behavior; another body may describe its structure; a third body may model the entity's operation in terms of register-transfer microoperations. There are no restrictions on the number of ways designers can view hardware entities.

- **INTERFACE DESCRIPTION**: A design entity's interface contains information common to its alternative bodies. A subset of this information (namely the specifications of *ports* and *generics*) is externally visible. The interface description of the system specifies the system's input and output (I/O) ports.

  Ports define communication channels between design entities and outside the world. A *port* definition involves description of its mode (*in, out, inout)* and type (like *bit, real, integer* etc...).

- **BEHAVIORAL DESCRIPTION**: In this modeling style, the behavior of the entity is expressed using sequentially executed, procedural code, which is very similar in syntax and semantics to that of a high level programming language like C or Pascal. We can write a pure behavioral description in VHDL using the *process statements.* A *process statement* is a concurrent statement which contains sequential statements that describe the functionality of a portion of entity in sequential terms. This process statement is sensitive to a set of signals defined by sensitivity list.

- **STRUCTURAL DESCRIPTION:** Structural descriptions are characterized by the use of subcomponents whose behavior is externally defined (in the form of another design entity). Such descriptions involve 1) *component declarations,* which declares the name and interfaces to subcomponents used in a design, and 2) *component instantiations,* which create one or more instances of a declared component. It associates the signals in the entity with the ports of that subcomponent.

- **PACKAGES:**This package concept was first introduced in the VHDL, no other HDL has this facility. All the technological dependencies are located into a unique location. i.e Package. So VHDL isolates technological dependencies from local definitions and this is a key feature, which makes VHDL very popular. We also use the package for making "type, constants and subprogram declarations" visible to all entity declarations and architectural bodies in the system design. If concept of package was not there then for each entity and architectural bodies we have to define type, constant and subprogram declarations. The general form of package declaration is:

  Package *identifier* is
  declaration 1;
  declaration 2;
  . . .
  . . .

  declaration N;
  end *identifier;*

The italic form of statements is to be filled up according to the user requirements.

- **LIBRARIES**: VHDL allows a designer to maintain multiple design libraries, and various parts of a single design can reside in different libraries. A library is used to store all the components of a given section of a design and can also be used in the design of another module, and thus it serves as a design re-use. Every VHDL simulator supports a WORK library which denotes the working library during analysis. The results of the analysis of a particular unit are stored in this WORK library for use in the next process.

- **MIXING OF BEHAVIORAL AND STRUCTURAL DESCRIPTION**: The behavioral and structural descriptions give you the functional interpretation of a physical system that you are modeling. The structural description gives you the total structure of your system with blocks and their interconnections.

- **MIXING OF CONCURRENT AND SEQUENTIAL BEHAVIOR**: Using VHDL we can create a single behavior containing both concurrent and sequential statements. In VHDL *process statements* can be placed wherever desired inside the block statements which is a concurrent mode and thus allowing creations where overall behavior is concurrent but some parts are described sequentially. Most of the HDL's do not support this method.

- **USER DEFINED TYPES**: VHDL allows users to define the data types and thus having a lot of freedom in the description of a system. Due to this feature we need not define behaviors using language pre-defined types and values, which may feel inconvenient to our design.

- **BUS RESOLUTION FUNCTIONS:** Bus resolution functions enable users to define what will happen to a signal when two sources are applied simultaneously when we face that conflict. In most of the HDL's this decision is defined in the algorithm and is not under the user control, but in VHDL this is under user control. For eg: In the following example *Signal_A* can get value 0 or 1 if *condition1* and *condition2* are true. So this can be decide by the user by writing a resolution function as shown in Fig :A-2

```
block
    signal Signal_A : out integer;
begin
    block begin
    Signal_A <= 0 when condition1 else
                x;
            ......
    block begin
    Signal_A <= 1 when condition1 else
                x;
            ......
end block;
```

Fig:AI-2 General from of resolution function

- **GENERIC PARAMETERS:** Use of *generic parameters* is very helpful during the design of a real system. We can change the timing information (*propagation delays, setup time, hold time...etc*) and physical information (*capacitance due to load, voltage levels, operating temperature etc...*).

- **SUPPORT FOR GRAPHICAL INFORMATION:** Initially the VHDL language has not supported graphical information, but after lot of research gone in this area, now it supports. This facility was built in the year 1993, the year on which the IEEE group has revised the VHDL features.

- A new attribute 'FOREIGN' is declared in VHDL in the year 1993 which is used in a sub-program to link in non-VHDL models.

- The operation of 'XNOR' was not available initially, but with the revision in the year 1993, the xnor operator is also available now.

- VHDL is tuned for multi-value logic modeling, and this can be done by overloading in VHDL. Overloading in VHDL allows us to change the meaning of the operator such as AND,OR etc. without having to change the VHDL model itself. For eg: '+' operator is supported only to add two real values, we can use ' ' as overloaded operator by declaring it to use to add a real and a integer value, two integers, two real values.

APPENDIX-II

Here a brief history of the evolvement of VHDL-AMS which is an extension of the VHDL, is given and then summarized it's important features that made the language put into use for analog/mixed signal circuits. VHDL-AMS denotes VHDL for Analog/Mixed Signal. This language had been recently standardized by the IEEE committee, and it is soon going to become an industry accepted standard language for analog/mixed signal circuits.

HISTORY OF VHDL-AMS:

Hardware Description Languages such as VHDL are today an essential technology to support most of the steps of digital hardware design, such as simulation, synthesis, testing and verification. As the IEEE 1076 standard, VHDL is committed to evolve through five years re-standardization cycles whose objective is to make the necessary language changes or extensions in response to feedback from users and from tool suppliers. The requirements that are needed to support analog and mixed signal systems have been issued during the initial phases of the second VHDL re-standardization cycle. Due to the complexity of the topic, a separate IEEE working group, referred as 1076.1, was formally formed in the year 1993 with the purpose to provide a language proposal based on VHDL 1076 that includes these new requirements. The language design phase is completed and a solid language architecture is defined. A formal IEEE balloting process to approve the proposal as the new IEEE standard 1076.1 has started in august 1997 and it has taken a very long process to get standardized and finally it got standardized on june 1999. Soon after VHDL-AMS was IEEE standardized companies like ANALOGY, MENTOR GRAPHICS & FTLSYSTEMS have tried to develop the VHDL-AMS simulators and these were available to the market in the end of december 1999 and early 2000.

This is an approach to mixed signal system simulation where the analog analysis routines and language processing functions were directly written in VHDL-87. This VHDL-AMS uses minimal language modifications to VHDL-93 to support analog system modeling. VHDL constructs are fully capable of handling signals in the continuous time domain but with slight syntax and semantics modifications. Interestingly VHDL has many characteristics which are equally applicable to both discrete and continuous mode modeling. Thus very little needs are added to VHDL to support mixed signal modeling. The 1076.1 language is a strict superset of VHDL 1076. The consequences are two fold. First, any legal VHDL 1076 description is also a legal VHDL 1076.1 model and provides the same simulation results when simulated with either a VHDL 1076 or a VHDL 1076.1 simulator. This also means that the new mixed-signal simulation cycle in VHDL 1076.1 reduces to the VHDL 1076 canonical simulation cycle where no analog part is present in the simulated model. Second the syntax and the semantics available in VHDL 1076 is reused as much as possible. As a result, some existing VHDL 1076 constructs are extended and new constructs are defined to support analog semantics. The approach 1076.1 is to provide a clear-cut distinction between "digital" and "analog" statements, while keeping the general VHDL philosophy. Lets discuss briefly some aspects in VHDL 1076.1.

1. *Behavior Aspects*:

Analog behavioral modeling is a key feature of VHDL 1076.1 as it allows the designer to develop his or her own models without having to rely on what a particular simulator is providing. It also allows a complete control on the abstraction level used for a particular model as well as on the characteristics of the system the model actually accounts for. The continuous aspects of the behavior of the lumped systems targeted by VHDL 1076.1 can be described by systems of ordinary differential and algebraic

equations with time as independent variable. VHDL 1076.1 provides a full support to the description of differential algebraic equation (DAE) in the time domain of the form

$$F(\dot{x}(t), x(t), t) = 0 \qquad \text{--(1)}$$

here F is a vector of linear or non-linear expressions, x is a vector of unknowns and $\dot{x}$ is a vector of first order derivatives of the unknowns w.r.t time and t is the time. The unknowns are usually physical quantities that depend on the discipline of the system being modeled. These physical quantities are voltage and current for electrical systems, velocity and force for mechanical systems, etc. The 1076.1 language defines the equations that are implied by the text of a model, however it doesn't define how these equations have to be assembled to form a system (1). This system is solved using analog solver (it is a conceptual representation of the agent that updates the values of the quantities of a model, and also updates the drivers of the implicit signals of the model.) which determines an analog solution point at time T when it determines these values for time T.

Quantities and *Terminals* are two new fundamental objects introduced in VHDL 1076.1 which find use in behavioral and structural description of analog and mixed signal systems. *Quantities* are value bearing objects that are evaluated in mixed-mode simulation whereas *Terminals* provide the structural interconnections for conservative systems described.

1.1 ***Quantities*:** *Quantities* belong to a new class of objects (like variable, signal etc.. in VHDL 1076) that represent the unknowns in the system(1). *Quantities* are piecewise continuous waveforms function of time that may only take floating-point values. *Quantities* get their values from the analog solver at specific times, the so called analog solution points (ASP's), that are determined by the analog solver. None of the other objects like signal, variable etc get their values the same way as *Quantities* get it. *Quantities* have to be declared before being referenced. They can be declared anywhere a signal declaration is allowed, except in a package. The *Quantities* can be declared as given below:

quantity *q1, q2* : *real* ;

For instance here entity declaration of a signal flow model of a two input adder with two interface quantities of mode in, and one interface quantity of mode out is shown in Fig: AII-1.

```
Entity adder is
        Port (quantity in1, in2 : in real ;
              quantity sum : out real );
end adder ;
```

Fig:AII-1 Entity declaration of an analog

The language also defines implicit *quantities*, i.e the *quantities* that are required in writing DAE's need not be declared.

- ***Q'Dot*** is an implicit quantity that is the first derivative of the explicit quantity Q w.r.t time *t*.
- ***Q'Integ*** is an implicit quantity that is the integral over time of quantity Q from time zero to the current time.
- ***Q'Delayed*(T)** is the value of quantity Q at a fixed interval T in the past (ideal delay).
- ***Q'ZOH*(T)** is the quantity Q sampled and held with sampling interval T.

1.2 ***Simultaneous Statements*** : These are new class of behavioral statements used to denote differential and algebraic equations that determine the values of the quantities of a model. The equations may be true input/output relationships or conservative branch equations. In the first case there is a directional computational flow and only one equation is allowed per output signal. In the second case, there is not any computational flow, but rather a set of equations that have to be solved simultaneously. The set of equations is built typically from the expression of KCL/KVL laws at connection points in which each branch equation contributes. These are allowed to be used in the statement part of the block, just like concurrent statements. The simultaneous statements can either be *simple simultaneous statements, simultaneous if statements, simultaneous case statements* and *simultaneous procedural statements*

☞ *Simple Simultaneous Statement* specifies zero or more characteristic expressions. The basic form of *simple simultaneous statement* has the following syntax:

Lhs_expression == Rhs expession ;

expressions *Lhs_expression, Rhs_expression* may be any legal VHDL numerical expression. The statement is symmetrical as both expressions may appear on any side. Quantities including implicit quantities (*Q'Dot, Q'Integ*) can be appeared on any side of the '= =' mark e.g: for capacitor

ic == C * cv'Dot ;

where ic, cv are current and voltage quantities and C is a constant which denotes capacitance value. e.g: for inductor

vl == L * il'Dot ;

where il, vl are current and voltage quantities and L is a constant which denotes inductance value.

☞ *Simultaneous if Statement* selects for evaluation one of the enclosed simultaneous statements parts depending on the value of one or more conditions. For the evaluation of a *simultaneous if statement*, the condition specified after if and after

elsif are evaluated in succession until one evaluates to true or all conditions are evaluated and yield false. A *simultaneous if statement* is a variant of a normal *sequential if statement* but is used in describing sets of equations that form part of the analog aspect of a model. A simultaneous if statement allows for simulation time replacement of equations in the system of equations. For example the behavior of MOSFET at its different regions can be modeled using simultaneous if statements as shown in fragment of code in Fig:AII-2

```
If vds >= 0.0 use
        If    (vgs - vt) <= 0.0  use
              ids == 0.0 ;
        elsif (vgs - vt) <= vds  use
              ids == 0.5 * beta * (vgs - vt)**2 ;
        else
              ids == beta * (vgs – vt - 0.5* vds)*vds ;
        end use ;
end use ;
```

Fig :AII-2. fragment of architecture behavior of

here vds, vgs are drain to source and gate to source voltages, ids is drain to source current, vt is the threshold voltage, beta is the current gain factor.

- *Simultaneous Case Statement* selects for evaluation one of a number of alternative simultaneous statement parts, the chosen alternative is defined by the value of expression. The expression of the simultaneous case statement, the expression of the choices of the simultaneous alternatives and the case label must follow the rules of the corresponding part of sequential case statement.

- *Simultaneous Procedural Statement* provides a sequential notation for expressing differential and algebraic equations. For example, consider the behavior of a weighted summer that which is governed by the following equation:

$$vout = \sum_{i=1}^{m} \beta_i * vinp - \sum_{j=1}^{n} \gamma_j * vinm \qquad \text{--(2)}$$

where vinp,vinm are postive and negative input voltages β, γ are the vectors of m positive weights and n negative weights, and vout is the output voltage.
equation(2) can be expressed by simultaneous procedural statement as shown in Fig:AII-3.

```
procedural
        variable x, y  : real : = 0.0 ;
        begin
        for i  in  beta'range   loop
                x : = x  +  beta(i) * vinp(i)  ;
        end loop;
        for j  in  beta'range   loop
                y : = y  +  gamma(j) * vinp(j)  ;
        end loop;
           vout  : = x  - y ;
end procedural ;
```

Fig :AII-3. fragment of architecture behavior of weighted

1.3 ***Tolerance groups*** : The quantities involved in simultaneous statements acquire their values from the analog solver. Actually the analog solver has to compute the values of quantities as to make the implied relations (like Lhs_expression – Rhs_expression) as close as possible to zero. To control this aspect and to remain unaffected with respect to simulation algorithms, the VHDL 1076.1 language introduces the concept of tolerance groups i.e. sets of quantities that have to meet the same accuracy requirements. VHDL 1076.1 allows to associate a string name tolerance code to tolerance groups. These tolerance codes can be addresses either directly or indirectly and this is illustrated as given below:

```
-- direct tolerance code assignment
     quantity cv : real  tolerance  ''min_voltage'' ;

-- indirect tolerance code assignment
     subtype voltage is real  tolerance  ''min_voltage'' ;
     quantity vl  : voltage ;
```

Fig :AII-4. tolerance code asignment

Implicit quantities like Q'Dot and Q'Integ belong to the same tolerance group of their parent quantity. The interpretation of the tolerance code is left to the simulator, i.e. the simulator environment must provide a way to associate tolerance codes to actual tolerances values that depend on the simulation algorithm in use.

1.4 ***Mixed-signal interactions*** : Mixed signal modeling and simulation involves the description and simulation of a model that contain parts whose behaviors are either digital or analog. VHDL 1076.1 allows the description of analog behavior that is

dependant on digital values and similarly the digital behavior that is dependant or sensitive to values of analog quantities.
A typical example of the first digital-to-analog kind of interaction is the reference of a signal name in a simple simultaneous statement.
e.g.: here quantity *q* and signal *s* are of type real, then *q* = = *s* introduces discontinuity in the vale of *q* whenever there is an event on *s*. So this discontinuity must be notified to the analog solver and is done by using break statement as shown below:

break *on s* ;

A typical example of the second analog-to-digital kind of interaction is a process that is triggered when a quantity crosses some threshold in the rising direction.
The following portion of code as in Fig:AII-5 shows the conversion of the value of the quantity q to the corresponding value of a two state logic value signal *s* :

```
process()
  variable high_level : real : = 5.0 ;
begin
  s <= '1' when q'above(high_level/2.0)
  else '0' ;
  ...
  ...
end process;
```

Fig:AII-5. A process for conversion of analog to digital

2.0 ***Simulation Aspects*** : VHDL 1076.1 extends the initialization phase and the simulation cycle as defined in VHDL 1076 to support mixed-signal initialization, quiescent state computation, user-defined initial conditions, mixed-signal time domain simulation, and frequency domain simulation.

2.1 ***Initialization and Initial Conditions*** : Initialization is a procedure that yields initial values of all signals and quantities in the model.. It defines the quiescent state of a mixed-signal model as the state at which there is no pending event at time zero. This includes the computation of the DC operating point for the analog part of the model. Initial values on signals or quantities are defined in their declaration. The default initial value on quantities is zero (a real number). Initial conditions are given as equations in the source code with a particular use of the break statement as shown in in the following syntax

break *q* => *expression* ;
break for *q1* **use** *q2* => *expression* ;

The initial value assignment is illustrated for the capacitor as shown in Fig:

```
-- direct way of initialization
    I == C * v'Dot ;
    Break  v => v0 ;
```

```
-- indirect way of initialization
        Q == C * v ;
        I == Q'Dot ;
        Break for Q use  v => v0 ;
```

In the above example the initial voltage v0 on capacitor is specified in two different ways.

2.2 ***Time Domain Mixed-Signal Simulation*** : In VHDL 1076 the kernel process is responsible to coordinate process activities during simulation and to update signal values accordingly. Time is considered as an integral multiple of some base unit (say 1fs) and time directly advances to the next event that is scheduled in the future. Similarly the analog solver solves the system of equations with a variable floating-point time step that is determined by the simulation algorithm in use.

Conversion functions are defined to convert between floating-point and physical time values with the equivalence as 1second of physical time is equal to 1.0 floating-point time. This mixed-signal simulation reduces to the existing simulation cycle if the model does not include any quantities, similarly it reduces to the single execution of analog solver if the model does not include any signals.

2.3 ***Small-Signal Frequency Domain Simulation and Noise Simulation*:**

The VHDL 1076.1 formally defines the small-signal model of system (1) as the linear incremental model obtained from its Taylor expansion. The resulting linear system of equations is then solved considering the application of frequency domain sources, which are defined in the text of the VHDL-AMS model as a special kind of quantities called *Source quantities.* These source quantities are of two types (a) *Spectral source quantities* (defines stimulus for small-signal AC simulation), and (b) *Noise source quantities*(defines stimulus for the noise simulaiton). These can be declared as shown in Fig:AII-6

```
-- spectral quantity declaration:
        quantity x : real spectrum  mag, phase ;
-- noise quantity declaration :
        quantity y : real noise  mag ;
```

Fig:AII-6 declaration of Spectral and Noise quantities

3.0 ***Structure Aspects*** : The description of analog and mixed-signal structures basically remains the same as that of digital structures, however for modeling of the interconnection of the analog components, new kinds of connection points are introduced.

3.1 ***Conservative Systems*** : A conservative system is modeled using a graph based approach involving two kinds of physical quantities called *across quantities,* and *through quantities.* Across quantities represent effects like voltage, temperature or

pressure, while Through quantities represent effects like current, heat flow rate or fluid flow rate.

3.2 ***Natures and Terminals*** : A nature defines the properties of a terminal as a pair of floating types. This is illustrated in the nature declaration shown below:

subtype *voltage,current* **is** *real* ;

nature *electrical* **is** *voltage* **across** *current* **through** ;

Similarly definitions can be made for other disciplines like mechanical, thermal etc.

A terminal does not bear any value by itself, and so unlike other objects it doesn't have nay type, but nature must be declared. For eg. Following declaration declares two terminals of nature electrical.

terminal *t1, t2* : *electrical* ;

A terminal can be declared anywhere a signal declaration is allowed.

Mathematical functions are not included in the 1076.1 language as they are available in the standard VHDL 1076.2 package called MATH_REAL and MATH_COMPLEX packages.

APPENDIX-III

The basic analysis of the switched capacitor Successive approximation Analog-to-Digital Converter is given in detail here. The circuit of ADC shown in Fig-3.3 reduces to simpler equivalent circuits depending upon the state of the digital signals like Sample ('S'),Hold ('H'),Φ_1, Φ_2 and Φ_3. From each equivalent circuit with simple analysis we can obtain the equations and this is shown already for states S=1,H=0,Φ_1=1,Φ_2=0,Φ_3=0; S=1,H=0,Φ_1=0,Φ_2=1,Φ_3=0; and S=1,H=0, Φ_1=0,Φ_2=0, Φ_3=1; The equivalent circuits for remaining possible states are shown here and corresponding equations are shown in detail.

S=0,H=1,b_I=0,Ø1=1,$Ø_2$=0,$Ø_3$=0

When sample signal is low (i.e. S='0') and hold signal is high (i.e. H='1'), the circuit functionality not only depends on the Φ_1,Φ_2 and Φ_3 signal states but also depends upon the state of the previous output bit b_i. Let's first consider the output bit b_I ='0'. Then equivalent circuit of digital-to-analog converter during the state of Φ_1 = '1', Φ_2 = '0' and Φ_3 = '0' is shown in Fig:AIII-1.

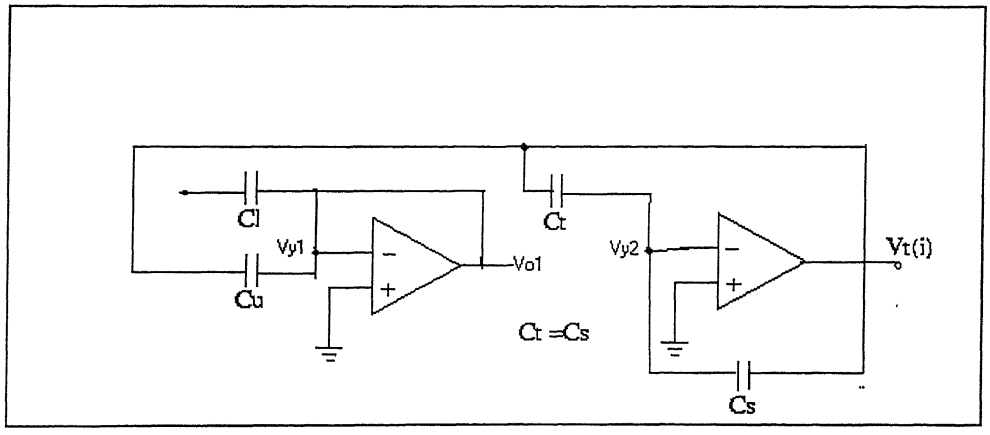


Fig:AIII-1. Equivalent circuit of DAC at b_i =0,S=0,H=1,Φ_1=1, Φ_2=0 and Φ_3=0.

The voltage across the capacitor C_u rises to the voltage across the capacitor C_t or C_s in the previous state. The voltage across capacitor C_l remains same. The voltages V_{y1}, V_{o1}, V_{y2} and V_{o2} are given as eq-(1) to eq-(3) . The voltage across capacitors C_t and C are equal and given by the eq-(4)

$$V_{y1} = V_{o1} = [-(AV_{off})/(1+A)] \quad (1)$$

$$V_{o2}=[\{(C_tA^2V_{off})/(1+A)^2\}+\{(C_tAV_r)/(1+A)\}+\{(C_sAV_{off})/(1+A)\}] [A/(1+A)][1/(C_t + C_s)] - [V_{off}(A/(1+A)] \quad (2)$$

$$V_{cu} = V_{y1}-V_{o2} \quad (3)$$

$$V_{ct} = V_{cs} = [\{-(C_tA^2V_{off})/(1+A)^2\} - \{(C_tAV_r)/(1+A)\}- \{(C_sAV_{off})/(1+A)\}][1/(C_t+Cs)] \quad (4)$$

From the equations in the previous state and equations in this state the voltage across capacitor C_l bears a simple relationship with that of V_{ct} in previous state as shown in eq-(5).

$$V_{cl} = [(AV_{ct})/(1+A)] \quad (5)$$

S=0,H=1,b_I=0,Ø1=0,$Ø_2$=1,$Ø_3$=0

Now when the next state i.e. Φ_1 = '0', Φ_2 = '1' and Φ_3 = '0' reaches with all other signals states remaining same i.e. b_I=0, S=0 and H=1 the equivalent circuit of analog-to-digital converter in this state is shown as in Fig:AIII-2.

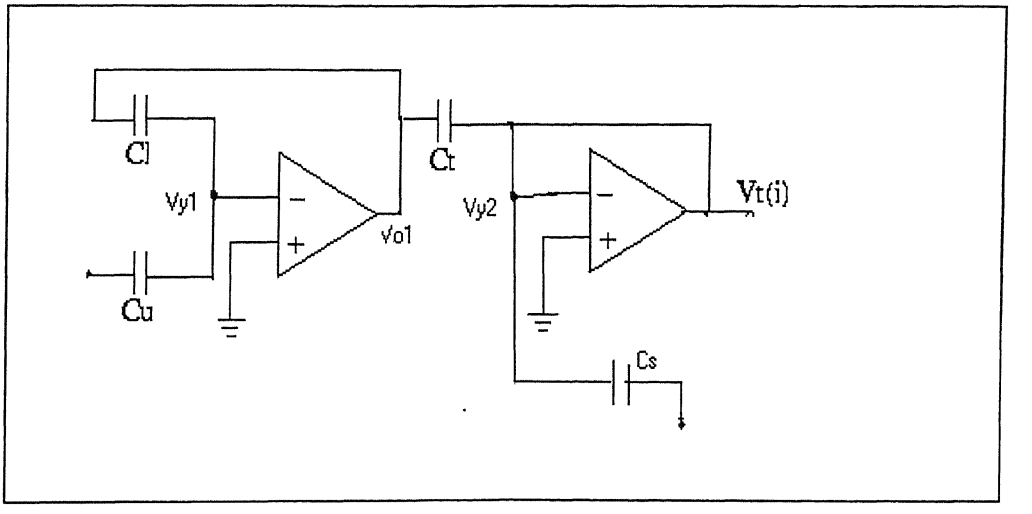


Fig:AIII-2. Equivalent circuit of DAC at b_i =0,S=0,H=1,Φ_1=0, Φ_2=1 and Φ_3=0.

The voltage across the capacitors C_u remains same as that of the previous state, similarly the voltage across capacitor C_s also remains same as that of its value in the previous state. The voltage across C_t rises to the V_{cl}. The voltage values of V_{o1},V_{o2},V_{cl} and V_{ct} are given as below:

$$V_{o1} = [(A^2V_{off})/(1+A)^2] - [(AV_{off})/(1+A)] \quad (6)$$

$$V_{y2} = V_{o2} = [-(AV_{off})/(1+A)] \quad (7)$$

$$V_{cl} = [-(AV_{off})/(1+A)] \quad (8)$$

$$V_{ct} = [-(A^2V_{off})/(1+A)^2] \quad (9)$$

From the equations in the previous state and equations in this state the voltage across capacitor C_t bears simple relationship with V_{cu} at previous state as shown in eq-(10).

$$V_{ct} = [-(AV_{cl}')/(1+A)] \quad (10)$$

S=0,H=1,b_I=0,Ø1=0,$Ø_2$=0,$Ø_3$=1

When the state Φ_1 = '0', Φ_2 = '0' and Φ_3 = '1' reaches the equivalent circuit of digital-to-analog converter at this state is same as shown in Fig:3.12. A divide by two operation occurs and finally the voltage across capacitors C_t and C_s are equal and are given by the eq-(11) and the output voltage V_t is given by eq-(12).

$$V_{ct}=V_{cs}=[-(C_tA^2V_{off})/(1+A)^2\}-\{(C_tAV_t)/(1+A)\}-\{(C_sAV_{off})/(1+A)\}]$$
$$[\{C_s/(C_t+C_s)^2\}] - [(A^2V_{off})/(1+A)^2] [C_t/(C_t+C_s)] \quad (11)$$

$$V_t=[\{[C_t/(C_t+C_s)][(A^2V_{off})/(1+A)^2]\}+[C_s/(C_t+C_s)][\{(C_tA^2Voff)/(1+A)^2\}$$
$$+ \{C_tAV_r/(1+A)\} + \{(C_sAV_{off})/(1+A)\}]][A/(1+A)] -[(AV_{off})/(1+A)] \quad (12)$$

Observing these equations and the equations in the previous state, we can simplify these equations in terms of previous voltage values as shown below:

$$V_{ct}=V_{cs}= [\ (C_tV_{ct}^{'}+ C_sV_{cs}^{'})/(C_t+C_s)\ ] \quad (13)$$

When the output bit b_I ='1', with sample signal remaining in active low state and hold signal at active high state, then the equivalent circuit of digital-to-analog converter during the state of Φ_1 = '1', Φ_2 = '0' and Φ_3 = '0' is same as shown in Fig:3.11 except that now the output V_t is connected to the C_l and not to C_u. This time the voltage across C_l rises to the previous value of V_{ct}. The exact equations can be obtained from simple analysis, and finally the voltage across capacitor C_l i.e. V_{cl} rises to the value of V_{ct} or V_{cs} in the previous state and is reduced to the eq-(14). The voltage across capacitor C_u remains same.

$$V_{cl} = [(AV_{ct}^{'})/(1+A)] \quad (14)$$

Now when the Φ_1 = '0', Φ_2 = '1' and Φ_3 = '0' state reaches then the equivalent circuit is same as shown in Fig:AIII-2 except that the output V_{ol} is connected to C_u instead of connecting to C_l. The voltage across capacitor C_t rises to the voltage across capacitor C_u in the previous state and is given simply by the eq-(15). The voltage across capacitor C_l remains same as that of its previous state.

$$V_{ct}= [(AV_{cu}^{'})/(1+A)] \quad (15)$$
$$V_{cs}= [-(AV_{off})/(1+A)] \quad (16)$$

Finally when it reaches the Φ_1 = '0', Φ_2 = '1' and Φ_3 = '0' state then the equivalent circuit of digital-to-analog converter is same as shown in Fig:3.12 and divide by two operation occurs, and the voltage across the capacitors C_t or C_s is given by the eq-(17).

$$V_{ct}=V_{cs}= [(C_tV_{ct}' + C_sV_{cs}')/(C_t+C_s)] \quad (17)$$

A 130798

Date Slip

This book is to be returned on the date last stamped.